Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations | 1. Thesis and Dissertation Collection, all items

1999-12

Design of a persistence server for the relational hypergraph model

Le, Hanh Cong Thi.

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/13447

Downloaded from NPS Archive: Calhoun

http://www.nps.edu/library

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN OF A PERSISTENCE SERVER FOR THE RELATIONAL HYPERGRAPH MODEL

by

Hanh Cong Thi Le

December 1999

Thesis Advisor: Valdis Berzins
Second Reader: Douglas Lange

Approved for public release; distribution is unlimited.

20000306 069

DTIC QUALITY INSPECTED 3

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1999	Master's Thesis

4. TITLE AND SUBTITLE
DESIGN OF A PERSISTENCE SERVER FOR THE RELATIONAL HYPERGRAPH MODEL

5. FUNDING NUMBERS

6. AUTHOR(S)
Le, Hanh C.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The fundamental purpose of this research is to develop an automated software evolution tool, CASES, for large and complex systems. CASES (Computer-Aided Software Evolution System) is based on the Relational Hypergraph model that is a formal model for describing software evolution processes. This model provides the preliminary mathematical definitions to support the development of CASES. There are five basic functions related to software evolution steps: step refinement, project evaluation, constraint management, personnel management, and step management. There are also five functions related to software evolution components: component management, component traceability, version control and configuration management, dependency management, and inference rule management. CASES is implemented by using Java JDK 1.1.7 and Swing 1.0.3 under the Visual Café version 3.0 environment. The primary contributions of this research include: (1) Providing an automated tool for software evolution; (2) Validating a software evolution model, the RH model; (3) Allowing reuse of software evolution components; (4) Describing the software evolution processes; (5) Automating the version control of software evolution objects; (6) Tracing the software evolution activities; and (7) Managing and controlling job scheduling and assignment.

14. SUBJECT TERMS
Software Evolution, Computer-Aided Software Evolution System (CASES)

15. NUMBER OF PAGES
293

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DESIGN OF A PERSISTENCE SERVER FOR THE RELATIONAL HYPERGRAPH MODEL

Hanh Cong Thi Le
B.S., San Diego State University, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1999

Author: [redacted]
Hanh Cong Thi Le

Approved by: [redacted]
Valdis Berzins, Thesis Advisor

[redacted]
Douglas Lange, Second Reader

[redacted]
Luqi, Chairman
Software Engineering Curriculum

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The fundamental purpose of this research is to develop an automated software evolution tool, CASES, for large and complex systems. CASES (Computer-Aided Software Evolution System) is based on the Relational Hypergraph model that is a formal model for describing software evolution processes. This model provides the preliminary mathematical definitions to support the development of CASES. There are five basic functions related to software evolution steps: step refinement, project evaluation, constraint management, personnel management, and step management. There are also five functions related to software evolution components: component management, component traceability, version control and configuration management, dependency management, and inference rule management. CASES is implemented by using Java JDK 1.1.7 and Swing 1.0.3 under the Visual Café version 3.0 environment. The primary contributions of this research include: (1) Providing an automated tool for software evolution; (2) Validating a software evolution model, the RH model; (3) Allowing reuse of software evolution components; (4) Describing the software evolution processes; (5) Automating the version control of software evolution objects; (6) Tracing the software evolution activities; and (7) Managing and controlling job scheduling and assignment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First of all, I would like to thank Dr. Luqi and Dr. Berzins who gave me a chance to work on CASES which helped me to learn much more about and gain a stronger understanding of Software Evolution.

I would like to say thank you to LTC Harn, who helped me to finish my master's thesis. It was very enjoyable and fun to work with LTC Harn.

Next, I will always remember the valuable advice from my understanding boss and appreciate all of his time spent and involvement in my thesis as the second reader, Mr. Lange at SPAWAR Systems Center San Diego.

Last, but not least, I thank God to give me all the luck and a wonderful family who has always been devoted to me and supported me whenever I needed it. Here, I would like to say "THANK YOU" a million times to each member of my wonderful family who helped and encouraged me to make my dream come true.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A software evolution tool, Computer-Aided Software Evolution System (CASES), has been designed and implemented for supporting the development of large and complicated software systems. Most automated tools similar to CASES, such as the INSERT system, created by CMU, and QSS DOORS, used by the U.S. Treasury [DATT99], have attempted to match the requirements of software evolution, but some essential problems of software evolution, like automated version control, component traceability, and job scheduling and assignment, are left unaddressed.

Roetzheim, [ROET98] stated that the United States has around 175,000 information projects to process, but that 31% of them are going to fail. Of these, 40% are canceled when it is recognized that their system design proves to be untenable. Another 33% are unfinished or are delayed as they run out of budget and time. Luqi [LUQI97] noted that formal methods can help industry, companies, and businesses improve these numbers. Many of the problems described in [ROET98] could be addressed by the integration of software evolution management and control, software evolution object reuse, and software evolution history tracing [HARN99f], which was the focus of this research.

The objective of this study was to design CASES. CASES can assist software evolution activities of software engineers and help them better control and manage the software evolution processes. In particular, CASES can provide a systematic and automatic environment for developing real-time embedded systems [HARN99d].

In order to accomplish this work, the following approaches were used: rapid prototyping [LUQI88a] [LUQI88b], formalization of software evolution objects [BADR94] [LIEB93] [SEIT98], and reuse of software objects [GOGU96] [HARN99a] [STEI91]. Rapid prototyping provides the best way to firm up the user's requirements. Formalization of software evolution objects provides the fundamental theorems of software evolution processes. The reuse of software objects enhances the reliability of software development.

CASES can provide automated assistance throughout software evolution processes and support the practical development of large complicated systems by physically distributed development teams. To describe and automate software evolution, the software evolution process is formalized using the Relational Hypergraph (RH) model. The RH model is an extension of the hypergraph model and evolutionary hypergraph model [HARN99b] [HARN99f] [LUQI97]. The RH model defines primary-input, secondary-input, and output dependencies among software evolution objects, and builds a multi-dimensional framework for software evolution processes [HARN99e].

The main functions of the RH model are to describe the software evolution objects and their dependencies, to formalize software evolution processes, and to construct the RH net for navigating software evolution objects.

CASES provides five functions related to the activities of software evolution: step refinement, project evaluation, constraint management, personnel management, and step management. CASES also provides five functions related to software evolution components: component management, component traceability, version control and configuration management, dependency management, and inference rule management.

CASES was built using object-oriented tools, Java JDK 1.1.7 and Swing 1.0.3 under the Visual Café version 3.0 environment [SYMA98]. CASES can interface to Microsoft Notepad, Microsoft Word, Microsoft Excel, and Netscape but also to a prototyping tool, Computer-Aided Prototyping System (CAPS) [LUQI88a]. CAPS is an easy to use, visual, and integrated tool that can be used to rapidly design real-time applications utilizing its prototype system description language (PSDL) editor, reusable software database, program generator, and real-time scheduler [BERZ97] [LUQI89] [LUQI93].

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATIONAL HYPERGRAPH MODEL

The RH model used in CASES is an extension of the graph model [LUQI90], hypergraph model [BERG89] [GALL93], and evolutionary hypergraph model [HARN99a] [LUQI97].

A. HYPERGRAPH MODEL

The hypergraph model represents the evolution history and future plans for software development as a hypergraph [LUQI97].

Definition 1. (hypergraph) A *(directed) hypergraph* is a tuple $H = (N, E, I, O)$ where

1. N is a set of *nodes*,
2. E is a set of *hyperedges* (*edges*),
3. $I : E \rightarrow 2^N$ is a function giving set of *inputs* of each hyperedge, and
4. $O : E \rightarrow 2^N$ is a function giving the set of *outputs* of each hyperedge.

This definition describes the bare structure of a hypergraph, where its nodes and hyperedges can be traced on paths of the hypergraph. Each path of the hypergraph represents an evolution history whose components, including nodes and hyperedges, can be traced.

B. EVOLUTIONARY HYPERGRAPH MODEL

The evolutionary hypergraph model, a kind of hypergraph, focuses on formalizing software evolution components and steps. Therefore, nodes and edges identify the software evolution components and steps respectively. The attributes of the components and steps must be recorded and labeled [LUQI97].

Definition 2. (Evolutionary Hypergraph) An *evolutionary hypergraph* is a labeled, directed, and acyclic hypergraph $H = (N, E, I, O)$ together with label functions $L_N : N \rightarrow C$ and $L_E : E \rightarrow A$ such that the following assumptions are satisfied:

1. The elements of N represent unique identifiers for software evolution components
2. The elements of E represent unique identifiers for software evolution steps
3. The functions I and O give the inputs and outputs of each software evolution step, such that $O(e) \cap O(e') \neq \varnothing$ implies $e = e'$
4. The function L_N labels each node with *component attributes* from the set C, including the corresponding version of the software evolution component
5. The function L_E labels each edge with attributes from the set A, including the current status of the software evolution step, such that $A = \{s, d\} \times A'$ (that is, each element of A has the form (s, a') or (d, a'), where $a' \in A'$.) and an edge labeled "*s*" is called a *step* and one labeled "*d*" is called a *decomposition*.

According to this definition, both components and steps can be refined into finer components and steps. In particular, the *minimal hypergraph* whose edge set has only one edge can also be refined into a finer hypergraph.

C. RELATIONAL HYPERGRAPH MODEL

A *relational hypergraph* [HARN99f] is an evolutionary hypergraph in which the *dependency* relationships between components and steps can have a hierarchy of specialized interpretations. These can be used to refine the software evolution process model.

For example, we can distinguish between primary-input-driven paths and secondary-input-driven paths. The input part to each hyperedge in a path could be a set of multiple input nodes containing many kinds of software evolution components. If there exist an input node and an output node to an evolutionary hyperedge, which are different versions of the same component, then the path from the input node via the hyperedge to the output node is called a *primary-input-driven path.* The hyperedge is called a *primary hyperedge*, and the relationship between the input node and the step is called a *primary_input dependency.* If there exists an input node and an output node to an evolutionary hyperedge, which are different components, then the path from the input node via the hyperedge to the output node is called a *secondary-input-driven path.* The hyperedge is called a *secondary hyperedge*, and the relationship between the input node and the step is called a *secondary_input dependency* [HARN99d].

Because input nodes to a step come from different entrances, a step in a relational hypergraph can be represented by an arrow with multiple tails, which are primary inputs

or secondary inputs as shown in Figure 6, or by combinational arrows shown in Figure 7 [HARN99f]. In order to describe traceability of the software evolution process, we use combinational arrows to illustrate paths of a step. The representation of combinational arrows points out a common output node and different paths (primary-input-driven paths or secondary-input-driven paths) of a step [HARN99d]. The structure of relational hypergraphs is an unlimited-depth hierarchy. In order to formalize the relationship between software evolution objects and software evolution tasks represented by steps, this research focuses on top-level and atomic-level relational hypergraphs. Some atomic-level objects are too big to be considered monolithic components, therefore, these objects must be decomposed. After they are decomposed, middle-level relational hypergraphs can be ignored because the software evolution objects in these levels are clusters of lower-level objects that represent temporary sets instead of representing the software evolution objects which can be carried out by stakeholders.

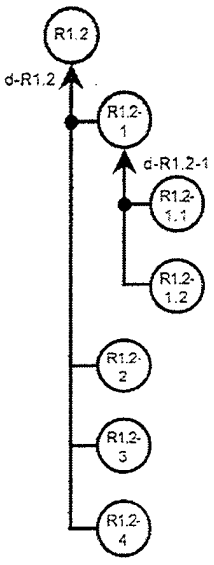


Figure 1: Three Layer Relational Hypergraph

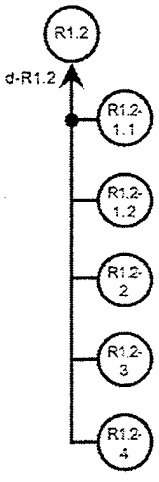


Figure 2: An Atomic Level Relational Hypergraph

In Figure 1 and 2 [HARN99d], node *R1.2* is a top-level requirement that includes five atomic-level requirements *R1.2-1.1, R1.2-1.2, R1.2-2, R1.2-3,* and *R1.2-4.* Figure 1 shows a three-layer relational hypergraph whose node *R1.2-1* is a middle-level requirement. Figure 2 is an atomic level representation of Figure 1.

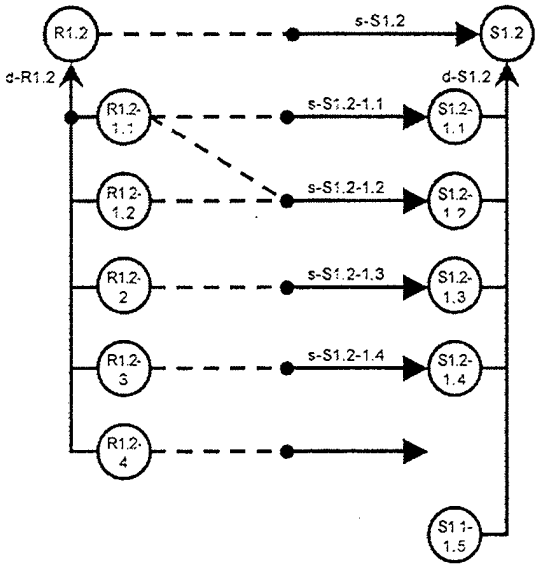


Figure 3: A Partial View of a Hypergraph(1)

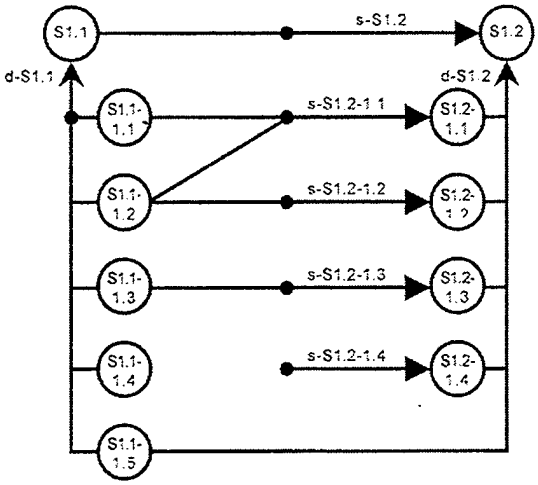


Figure 4: A Partial View of a Hypergraph(2)

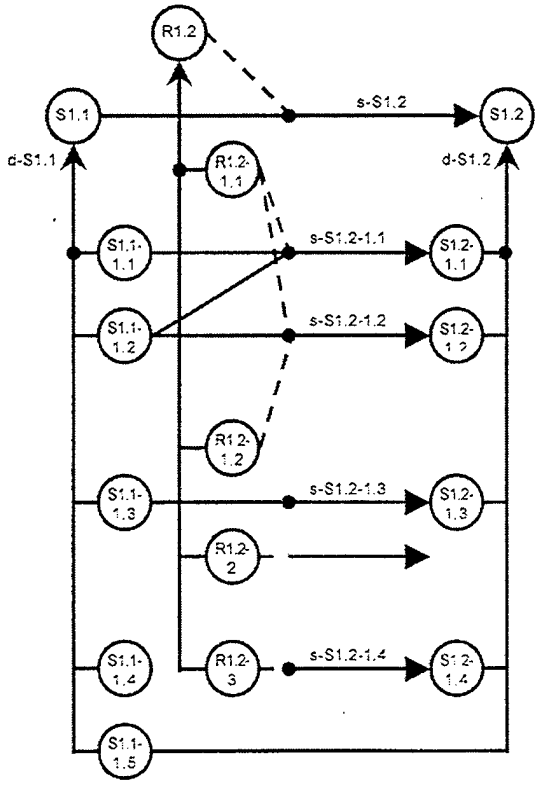
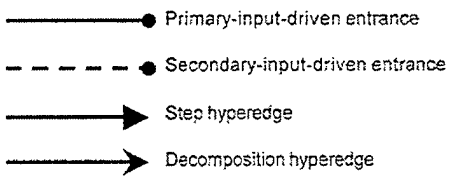


Figure 5: A Relational Hypergraph



Each input to a step *s* is individual and can be connected to others. For example, inputs *R1.2* and *S1.1* to the step *s-S1.2* in Figure 3 and 4, respectively, are individual inputs and can be connected to each other as shown in Figure 5. Figures 3 and 4 are partial views of Figure 5. This feature can be applied to both top-level and atomic level steps [HARN99d].

A software evolution step with its input and output components forms a Step Processed In Different Entrance Relationship, (SPIDER) which simplifies the dependency complexity of the software evolution and helps to construct the relational hypergraph net, shown as Figure 6 and Figure 7. Each top-level, refined, and atomic level SPIDER is connected into a relational hypergraph net. An atomic level SPIDER is a minimal task unit that can be assigned to developers.

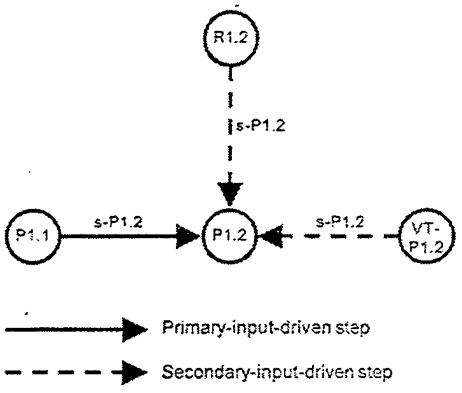


Figure 6: SPIDER(1)

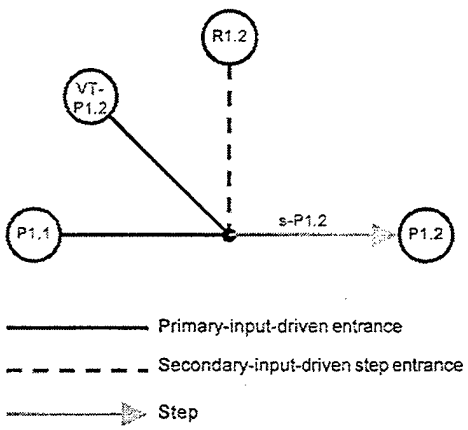


Figure 7: SPIDER(2)

D. FORMALIZED SOFTWARE EVOLUTION PROCESS

According to the Schematic Model of the Analysis Process [BERZ97] modified from the IBIS (Issue-Based Information System) model [CONK88], the software evolution process has identified eight types of top-level steps: software prototype demo,

issue analysis, requirement analysis, specification design, module implementation, program integration, software product demo, and software product implementation. The software evolution process has also identified eight different types of top-level components: criticisms, issues, requirements, specifications, modules, software prototype programs, optimizations, and software product programs. Each top-level object can be decomposed into a set of atomic objects, either directly or indirectly. This formalization of the modified software evolution process obtains user requirements via software prototype demo, issue analysis, and requirement analysis steps as shown in Figure 8 [HARN99f]. Typically, the software evolution processes can be changed according to development environment needs.

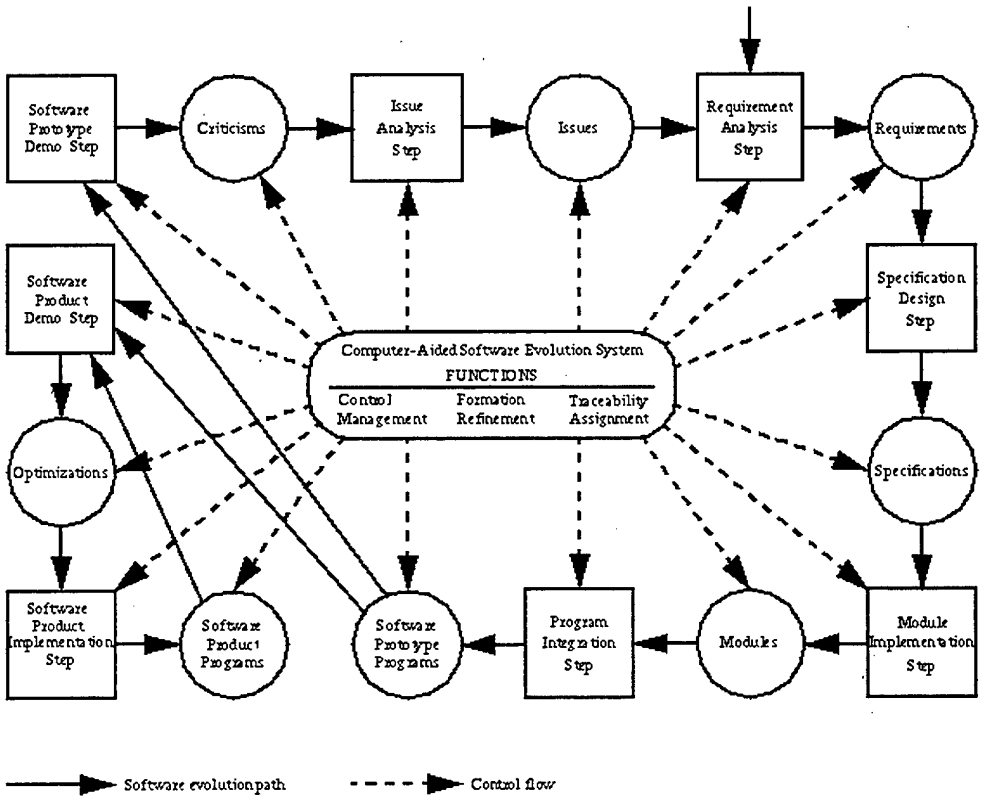


Figure 8: Software Evolution Processes with CASES

THIS PAGE INTENTIONALLY LEFT BLANK

III. CASES DESIGN

CASES users include project managers or leaders, project organizers, project evaluators, system designers, and system analysts. CASES provides an interface to connect many kinds of application tools. In this research, CASES can link to Microsoft Notepad, Microsoft Word, Microsoft Excel, Netscape, and CAPS application. The system context diagram shown in Figure 9 describes the relationships between CASES and outside objects [HARN99f].

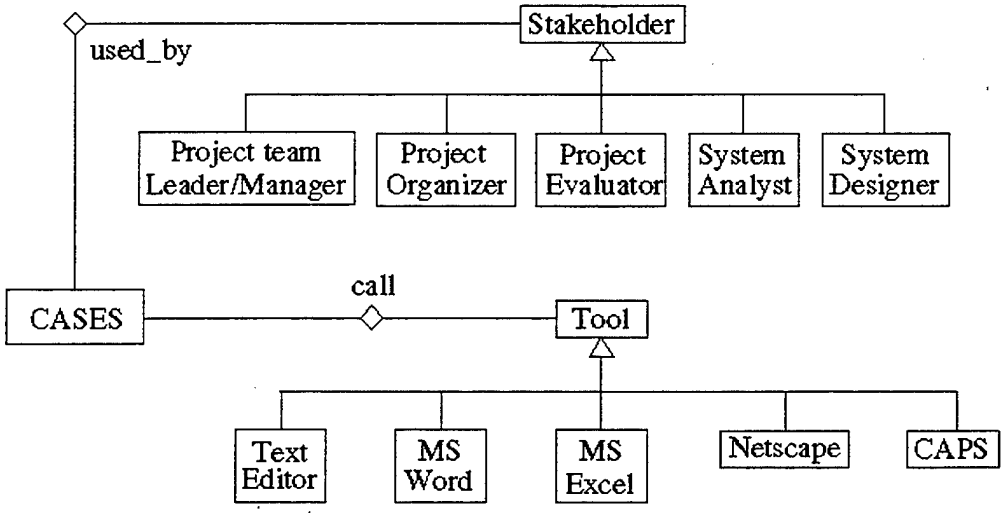


Figure 9: Context Diagram of CASES

There are five fundamental functions in CASES: project schema management, automated version control, SPIDER formation and tracing, tools, and job scheduling and management [EVAN97]. In project schema management, CASES helps users to propose software development processes and to determine related software evolution steps, components, and dependencies. CASES can automatically create new variant and version numbers for steps in a specified software evolution process. This research also

designed the software evolution history splitting and merging for switching variant paths of software evolution history. In the SPIDER formation and tracing, CASES users can edit, decompose, and trace SPIDERs. We also provide functions to record step and component contents. CASES users navigate related SPIDERs backwards or forwards without any limitations and can view the content of a step and component. CASES has five connections to communicate with outside software packages. Beside the connections, CASES has a function to manage personnel data. Finally, CASES can assign and schedule jobs for a person in the personnel database of the *stakeholder* directory.

CASES was built using the Visual Café version 3.0 environment, and it includes Java Development Kit (JDK) 1.1.7 and Swing 1.0.3 in the Visual Café library, where Swing classes are intended to make Graphical User Interfaces (GUIs). CASES is platform independent, which can run under Unix, Windows NT, or Windows 95. The only requirement needed to run CASES is to install JDK1.1.7 and Swing1.0.3 in the system.

A. ARCHITECTURE OVERVIEW

CASES uses the file structure of the Java system to save and to retrieve data. CASES creates objects and saves them in files with the benefit of getting whole objects. All objects saved in files are serialized objects since the serialized object is an instant file format that works for any application[FLAN97].

1. The CASES Package

CASES is structured as one package with thirty-five Java files. Most files in this package are reusable through their class' construction and launched by the *CasesFrame.java* file. In the CASES design, files are reusable to avoid the necessity of creating multiple files for a similar purpose. This design requires less hardware space and memory to execute and will therfore run more quickly and efficiently than if files were not reusable.

The use of global variables saves time in running CASES. The file *CasesTitle.java* is the only file which stores all the global variables for CASES and all titles for the whole system. This makes it simple to change or update titles, filenames, numbers, lists, or locations. Most of the Java files in this package implement *CasesTitle.java*. To change a title or directory, one only needs to go to *CasesTitle.java* and modify the directory path from there.

Interface classes require another file for implementation and those functions must be included in an implemented class. Programmers need only observe the code in CASES interface classes to know its content and also what kind of structure is needed to extend a model. *CasesFrame.java* is no less important since it is the main driver for CASES. *CasesFrame.java* is not an inherited class, but is the main connection required to launch other classes.

2. Architecture

CASES is not a complicated system, however it is difficult to describe its architecture by the use of words. The following *Rational Rose* diagram, Figure 10, will help to explain the architecture of CASES. This class diagram of CASES only shows

associations among classes, since no classes are composed of others. *CasesFrame.java* is needed as the main bridge for connections.

This class diagram does not list the all minor and interface class files in the CASES package. It only shows the main connections among them.

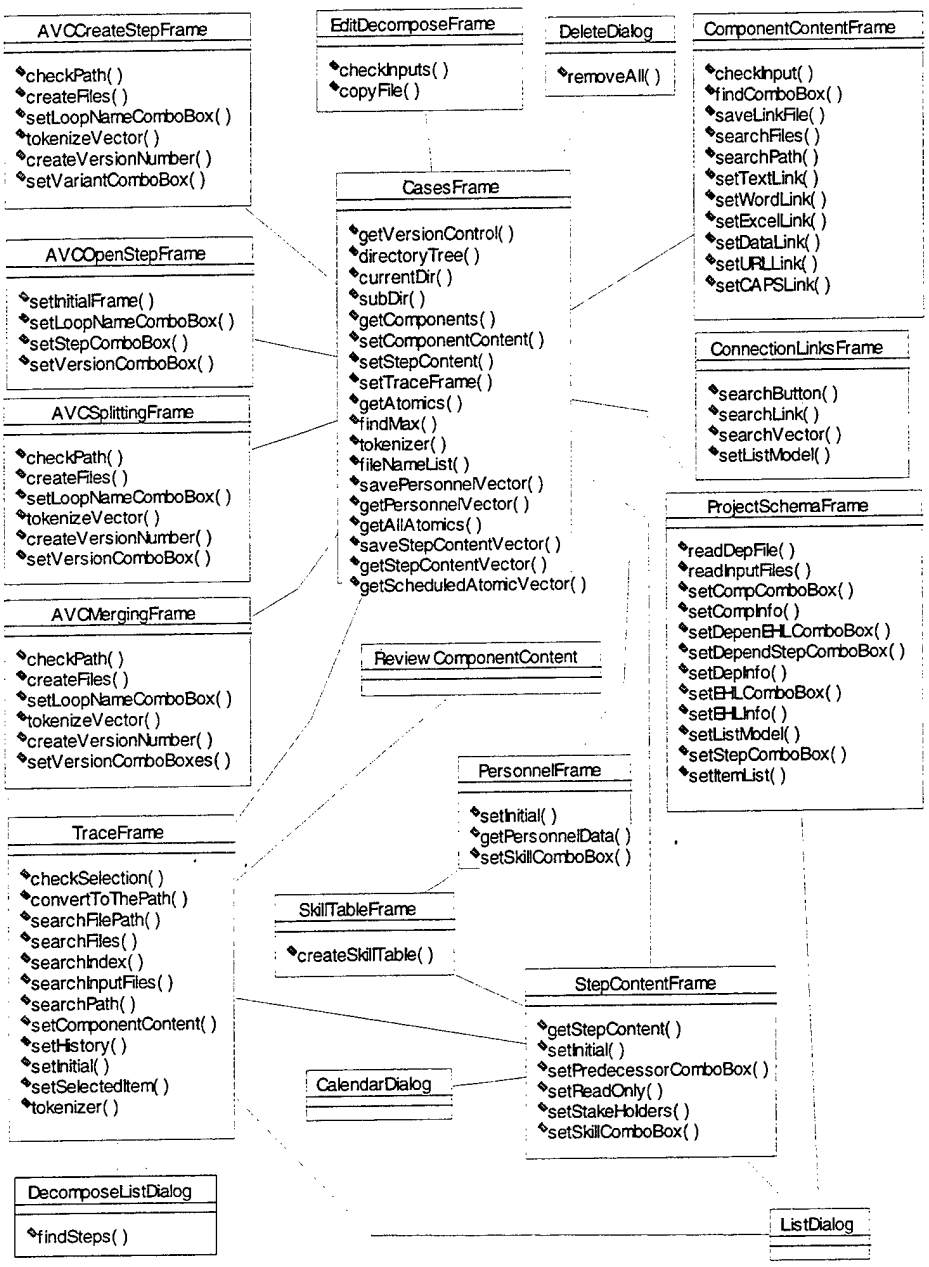


Figure 10: CASES Diagram

B. FILE STRUCTURE AND DEPENDENCY RULES

The file structure of CASES is built to match the multi-dimensional architecture of the Relational Hypergraph (RH). According to the evolutionary hypergraph model, a step has a unique output component [LUQI97]. The identifier of this component is inherited from its source step. For example, if the identifier of a step is *s-R1.1*, then the identifier of output component of this step is specified to *R1.1* through the automated version control of CASES. Without the file structure, CASES cannot trace software evolution steps and components forward and backward. The diagram shown in Figure 11 [HARN99f] describes the file structure of CASES that includes the CASES directory, project names, step identifiers and related files, version numbers, and SPIDER construction.

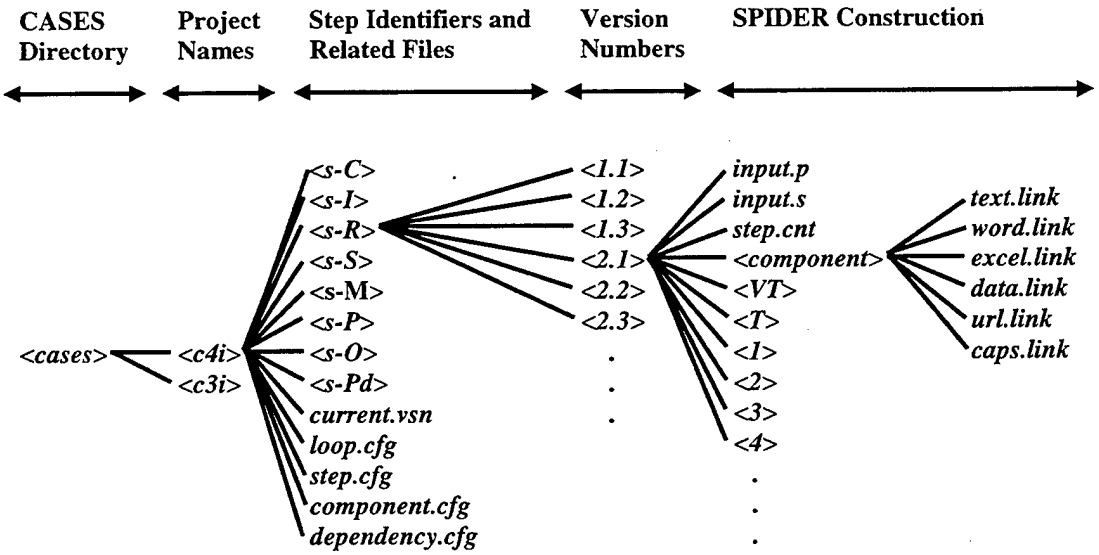


Figure 11: CASES File Structure

CASES creates new components and new version numbers without duplicating existing files in the file structure. To fulfill this requirement, CASES has algorithms and

dependency rules [BERZ98] [HARN99c] [HARN99f] to automate version control. For example, the variant and version numbering rules are as follows:

Rule 1:

Let *c1* and *c2* denote two different components and *s* denote a step. For all *c1*, *c2*, and *s*, it is the case that if the dependency of *c1* and *s* is primary_input, the dependency of *c2* and s is output, and the dependency of *c1* and *c2* is *same_variant*, then the version number of *c2* is equal to the version number of *c1* plus 1.

ALL(*c1 c2* : C, *s* : S :: *c1* primary_input *s* & *c2* output *s* & *c1 same_variant c2* => *c2.version.version_number* = *c1.version.version_number* + 1).

Rule2:

Let *c1* and *c2* denote two different components and s denote a step.. For all *c1*, *c2*, and *s*, it is the case that if the dependency of *c1* and *s* is primary_input, the dependency of *c2* and *s* is output, and the dependency of *c1* and *c2* is *used_by.split*, then the version number of *c2* is equal to the version number of *c1* plus 1 and the variant number of *c2* is equal to the highest variant number of object *c1* plus 1.

ALL(*c1 c2* : C, *s* : S :: *c1* primary_input *s* & *c2* output *s* & *c1 used_by.split c2* => *c2.version.version_number* = *c1.version.version_number* + 1 & *c2.version.variant_number* = *highest_variant_number*(*c1.version.object_id*) + 1).

Rule3:

Let *c1*, *c2*, and *c3* denote three different components and *s* denote a step. For all *c1*, *c2*, *c3*, and *s*, it is the case that if the dependency of *c1* and *s* is primary_input, the

dependency of *c2* and *s* is primary_input, the dependency of *c3* and *s* is output, and the dependency of *c1* and *c2* is *used_by.merge.new_variant*, then the version number of *c3* is equal to the maximum version number of *c1* and *c2* plus 1 and the variant number of *c3* is equal to the highest variant number of object *c1* plus 1.

ALL(*c1 c2 c3*: C, *s* : S :: *c1* primary_input *s* & *c2* primary_input s & *c3* output *s* & *c1* *used_by.merge.new_variant* *c2* => *c3.version.version_number* = max(*c1.version.version_number*, *c2.version.version_number*) + 1 & *c3.version.variant_number* = *highest_variant_number*(*c1.version.object_id*) + 1).

Rule4:

Let *c1*, *c2*, and *c3* denote three different components and *s* denote a step. For all *c1*, *c2*, *c3*, and *s*, it is the case that if the dependency of *c2* and s is primary_input, the dependency of *c3* and *s* is output, the dependency of *c1* and *c2* is *used_by.merge.old_variant*, and the dependency of *c1* and *c3* is *used_by.split*, then the version number of *c3* is equal to the maximum version number of *c1* and *c2* plus.

ALL(*c1 c2 c3*: C, *s* : S :: *c2* primary_input *s* & *c3* output *s* & *c1* *used_by.merge.old_variant c2* & *c1 used_by.split c3* => *c3.version.version_number* = max(*c1.version.version_number*, *c2.version.version_number*) + 1).

To create new variant and version numbers, the *Automated Version Control — Create Step* of CASES has to increase a variant and version number by 0.1 from the previous variant and version number. In the software evolution splitting case, the *Automated Version Control — Evolution History Splitting* of CASES, based on the

highest variant number of all the steps in the selected process, increases this number by 1 and increases the selected version number by 0.1 [DAMP94] [BADR94].

Similarly, *Automated Version Control — Evolution History Merging* has its own rules, which results in increasing the maximum variant and version number by 1 and 0.1 respectively if the new component is merged to new variant, or results in increasing the selected variant and version number by 0 and 0.1 respectively if the new component is merged to the same old variant[DAMP94] [BADR94].

CASES also has to copy the files, *input.p* and *input.s*, and the directory, *Component Content* from the closest top level to the new refined level. This means that the file structure of the new refined level will inherit files from its parent. CASES provides edit functions for users flexibly to modify the inherited file names.

Above rules can be implemented by the following code:

Rule 1:

```
String sub1 = ((String)currentVariantComboBox.getSelectedItem()).trim();
String sub2 = list[0].substring(index+1);

int theMax = Integer.parseInt(sub2);

for( int i=1; i<list.length; i++ ){
        index = list[i].indexOf(".");
        String s = list[i].substring(0, index);
        if( s.equals(sub1)){
                sub2 = list[i].substring(index+1);
                int temp = Math.max(theMax,Integer.parseInt(sub2));
                theMax = temp;
        }
}
theMax++;
String stepVersion =
((String)this.currentVariantComboBox.getSelectedItem()).trim()+"."+theMax;
```

Rule 2:

```
int i1 = Integer.parseInt(sub1)+1;
int i2 = Integer.parseInt(sub2)+1;
```

```
String stepVersion = i1+"."+i2;

for( int i=0; i<currentStepComboBox.getItemCount(); i++ ){
        String s = (String)currentStepComboBox.getItemAt(i);
        ...
        else if( i==currentStepComboBox.getItemCount()-1){
                newVersionTextField.setText(stepVersion);
        }
}
```

Rule 3:

```
int index1 = current.indexOf(".");
int index2 = merged.indexOf(".");

String mergedVersion = null;
String sub1 = current.substring(index1+1);
String sub2 = merged.substring(index2+1);

int theMax2 = Math.max(Integer.parseInt(sub1),Integer.parseInt(sub2));
theMax2++;

if( typeIndex == 0 ){
        mergedVersion = current.substring(0,index1)+"."+theMax2;
}
```

Rule 4:

```
int index1 = current.indexOf(".");
int index2 = merged.indexOf(".");

String mergedVersion = null;
String sub1 = current.substring(index1+1);
String sub2 = merged.substring(index2+1);

int theMax2 = Math.max(Integer.parseInt(sub1),Integer.parseInt(sub2));
theMax2++;
...
else if( typeIndex == 1 ){
        index1 = current.indexOf(".");
        index2 = merged.indexOf(".");
        sub1 = current.substring(0, index1);
        sub2 = merged.substring(0, index2);
        int theMax1 = Math.max(Integer.parseInt(sub1),Integer.parseInt(sub2));
        theMax1++;
        mergedVersion = theMax1+"."+theMax2;
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASES IMPLEMENTATION

CASES is a friendly tool for software evolution. The implementation of CASES is based on the multi-dimensional architecture of the RH model. Most importantly, CASES can save and retrieve the RH model without using any database tools.

CASES consists of thirty-five Java files which accomplish object serialization, establishment and control of interfaces, menus and applications.

A. OBJECT SERIALIZATION

Serialization of objects is one of the most important mechanisms and improvements in Java 1.1, since it allows the creation and editing of any type of custom-built objects stored in its files. By using this powerful capability, objects can easily be written in and read back from input and output streams. These object streams can be saved into files and their object types, and complete content can be recovered from these files [FLAN97].

Taking advantage of the power of object serialization, CASES creates seven object streams. They are saved and used in files designated as: *step.cfg*, *component.cfg*, *loop.cfg*, *current.vsn*, *step.cnt*, and *dependency.cfg*. *Personnel* objects are saved in the *stakeholder* directory. Each one of these seven object streams has its own functionality. Table 1 describes their functionality and attributes in CASES.

Object	Attribute	Functionality	Example
Component Type	Identification (ID)	ID of this component type object	R
	Name	Name of this component type object	Requirement
	Description	Description of this component type object	Any remark
Dependency	Process Name	Process name of this dependency object	Software Product Generation Process
	Step Name	One of step names in the process path	s-R
	Output Component	Output of the selected step	R (default)
	Primary Input	Primary input of this dependency object	R (default)
	Secondary Input	Secondary input of this dependency object.	I (one of available component types in this project
EHL	EHL Name (Process Name)	Name of process	Software Product Generation Process
	EHL Path (Process Path)	Contains all selected step types	s-C, s-R, s-I
Personnel	Identification (ID)	ID of the personnel object	3015
	Name	Name of a person	Mr. Smith

Table 1: CASES Object

Object	Attribute	Functionality	Example
	Skill	Skills of this person	3 : TAE plus : 1 6 : Ada : 3
	E-Mail	E-Mail address of this person	Smith@cs.nps.navy.mil
	Telephone	Telephone number of this person	(831) 656-2000
	Fax	Fax number of this person	(831) 656-3000
	Address	Address of this person	2000 NPS Blvd. Monterey, CA 40000
Step Content	Step Version	Step name and version number of this step	s-R1.2-2
	Status	Status of this step content	Approved
	Skill	Skill required for this step	2 : CAPS : 3 10 : Rational Rose : 2
	Security Level	Security level of this step	3
	Evaluation	Evaluation of the performance of this step	Work-alcoholic
	Evaluator	ID of this step's evaluator	3215
	Organizer	ID of this step's organizer	3508
	Predecessors	Contains predecessors of this step	s-R1.1-1.5 s-r1.1-2.1

Table 1: CASES Object

Object	Attribute	Functionality	Example
	Priority	Priority of this step	3
	Estimate Duration	Predict how long this step will finish	10
	Earliest Start Time	Can not start this step before this day	June 30, 1999
	Finish Time	Exact day to finish this step	October 31, 1999
	Manager	ID of this step's manager	2467
Step Type	Identification (ID)	ID of this step's type object	s-R
	Name	Name of this step type object	Software Product demo step
	Description	Description about this step type object	Any remark
Version Control	Current Process	The current working process' name	Software Product Generation Process
	Current Step	The current working step's name	s-R
	Current Version	The version of the current step	1.2-2
	Current Status	The status of the current step	Approved

Table 1: CASES Object

B. MENUS

The menu bar in CASES contains five menus: *Project, Automated Version Control*, *SPIDER*, *Tools*, and *Job Schedule*. Each menu holds its own menu items. Each menu item is used to launch a specific application. When CASES is first launched, all, except *Project* menu, are grayed out, since no project is selected yet. The *Open* and *Delete* sub-menus of *Project* are also grayed out if the *CASES* directory is empty.

All of the menu items in the *SPIDER* menu specify restriction conditions before launching applications. For example, a menu item of *SPIDER* menu will display an error message if a *current.vsn* file does not exist in the *CASES* directory, since CASES can not automatically define the step the CASES user wants to process.

On the other hand, the menu items of *Automated Version Control* do not restrict launching applications. Applications may not exist, if *loop.cfg* and *dependency.cfg* are empty. This situation will only happen if the CASES user does not create *Step Type*, *Component Type*, *Evolution Process*, and *Dependency* for the *step.cfg*, *component.cfg*, *loop.cfg*, and *dependency.cfg* files. For this reason *ProjectSchemaFrame* is automatically launched whenever a user creates a new project.

If the CASES user opens an existing project, CASES allows the user to *add*, *delete*, or *edit Step Type*, *Component Type*, *Evolution Process*, and *Dependency* data. More details of CASES menus are shown in Table 2.

Menu	Menu Item	Functionality	Restrictions
Project	Create Project	Prompt to input new project name and create a project	No restriction
	Open Project	Open one of the existing projects under CASES directory	At least one project exists in CASES directory
	Delete Project	Prompt to select an existing project name and delete the entire project	At least one project exists in CASES directory and the project is not open
	Exit	Exit CasesFrame	No restriction
Automatic Version Control	Create Step Version	Create new version number for all steps in the selected process	No restriction, however, more productive if step type, component type, and evolution process were created
	Open Step Version	Select a step and its version number. They will be saved in current.vsn file as a current step to process.	No restriction, however, more productive if step type, component type, and evolution process were created
	Evolution History Splitting	Create the new version number for the entire steps in the selected process which is based on the selected version	No restriction, however, more productive if step type, component type, and evolution process were created
	Evolution History Merging	Create the new version number for the entire step in the selected process which is based on the current version and the selected version number	No restriction, however, more productive if step type, component type, and evolution process were created
SPIDER	Edit	Edit input.p, input.s files and Component Content directory	User must select a component since CASES only allow to edit the selected component and its children
	Decompose	Decompose the selected step into the lower version	User must select a component since CASES only allow to edit the selected component and its children

Table 2: CASES Menu

Menu	Menu Item	Functionality	Restrictions
	Component Content	Create link files under Component Content directory	User must select a component since CASES only allow to edit the selected component and its children
	Step Content	Create step.cnt file for the selected step	User must select a component since CASES only allow to edit the selected component and its children
	Trace	View content of steps	User must select a component since CASES only allow to edit the selected component and its children
Tools	Text	Launch notepad application and allow user to create a text file	No restriction
	MS Word	Launch MS Word application and allow user to create a word file	No restriction
	MS Excel	Launch MS Excel application and allow user to create an excel file	No restriction
	Netscape	Launch Netscape application	No restriction
	Caps	Launch Heterogeneous Systems Integrator and allow to create PSDL file	No restriction
Tools-Personnel Data	Add	Launch PersonnelFrame and allow user to create Personnel object	No restriction
	Edit	Launch PersonnelFrame and allow user to create Personnel object	No restriction
	Delete	Launch DeleteDialog and allow user to select any personnel object in stakeholder directory to delete	No restriction
Job Schedule	Scheduling	Launch Scheduling application	
	Assignment	Launch Scheduling application	

Table 2: CASES Menu

C. INTERFACES

Unlike C++, Java does not have multiple inheritance capability. Java only extends one super class and allows the addition of more than one interface in the class. The key word *interface* in Java's object-oriented style is used to define a class that the user wants to be inherited [FLAN97].

Whenever a class implements an interface, the class has to define all the methods which are listed in the interface. *Interface* plays a role in such a commitment. When a class implements an interface, that class must complete the to-do-list of these methods. The class has to define these methods with exactly the same signature in the interface. However, the key to implementing an interface class in Java is that the interface does not need to know how these methods are defined in the class as long as the class has the same signatures as the interface.

CASES has nine interfaces with the purpose of advising the future programmers to include these methods in their classes when they implement interfaces (See Table 3).

Interface	Implemented By	Functionality
I_Cases	CasesFrame	Interface to create main frame for CASES
I_AVC	AVCCreateStepFrame AVCSplittingFrame AVCMergingFrame	Interface to create new step version Interface to split step version Interface to merge step versions
I_AVCOpenStep	AVCOpenStepFrame	Interface to open step version
I_ComponentContent	ComponentContentFrame	Interface to create/view/edit/delete link files in the Component Content directory
I_EditDecompose	EditDecomposeFrame	Interface to edit/decompose the current step
I_Personnel	PersonnelFrame	Interface to create/edit/view a personnel object
I_ProjectSchema	ProjectSchemaFrame	Interface to create/edit step.cfg, component.cfg, loop.cfg, and dependency.cfg
I_Trace	TraceFrame	Interface to view the selected component

Table 3: CASES Interface

D. APPLICATIONS

The majority of Java files in CASES are applications. These applications include *JFrame*, *JDialog*, *JOptionPane*, *FileDialog*, and *JFileChooser* (shown in Table 4).

JFrame allows the user to input information and to view the output from a file. *JDialog* has the same behavior as *JFrame* but has an advantage in that *JDialog* does not allow a user to open the next application until *JDialog* has been exited.

JOptionPane has a different purpose. It is unnecessary to use *JFrame* or *JDialog* if the CASES user only needs to input one string such as the project name. *JOptionPane* has the input functionality to provide that option. CASES uses *JOptionPane* to create a new project since CASES only needs a new project name in order to initiate the process. *JOptionPane* also provides warning messages when errors occur, confirmation messages in order to continue, and queries to save, or upon exit.

FileDialog and *JFileChooser* perform similar funtions which show the system file structure. Both are used in CASES based on their characteristics and the specific requirement needed. *FileDialog* has a more attractive application interface compared to *JFileChooser*, however, *FileDialog* has the disadvantage of forcing the user to select a file instead of a directory before continuing the other functions.

FileDialog is the only application that is a direct sub-class of the *java.awt* package. The other applications are derived from *Java Foundation Classes* (JFC). All of the outputs of CASES applications are listed in Appendix A.

Application	Launched by	Functionality
CasesFrame	Self	Main frame of CASES
AVCCreateStepFrame	CasesFrame	Create new step version
AVCOpenStepFrame	CasesFrame	Open one of existing step versions to process
AVCSplittingFrame	CasesFrame	Splitting the selected step version into the new step version
AVCMergingFrame	CasesFrame	Merging the selected steps into the new step version
EditDecomposeFrame	CasesFrame	1) Edit input.p and input.s files of the current step, or 2) Decompose the current step into the new version and allow to edit input.p and input.s files of the current step.
ComponentContentFrame	CasesFrame, TraceFrame	Create link files under Component Content directory of the selected step
StepContentFrame	CasesFrame, TraceFrame	Create step.cnt file in the selected step
TraceFrame	CasesFrame	View contents of the selected step
PersonnelFrame	CasesFrame, TraceFrame, ReviewComponentContentDialog	Create/View personnel object

Table 4: CASES Application

Application	Launched by	Functionality
ProjectSchemaFrame	JoptionPane	Create/Edit step.cfg, component.cfg, loop.cfg, and dependency.cfg files
ReviewComponentContentDialog	TraceFrame	View component content of the selected step. It will use the content of links file as a parameter to connect to their equivalent application, e.g., notepad, MS Word, MS Excel, Netscape, Caps, or PersonnelFrame
ListDialog	StepContentFrame, TraceFrame, ProjectSchemaFrame	List atomics, components, or the path of these components
DeleteDialog	CasesFrame	Delete a personnel object
DescomposeListDialog	TraceFrame	List decompose components of one step at a time
SkillTableFrame	StepContentFrame, PersonnelFrame	List skill ID, skill name, and skill level
CalendarDialog	StepContentFrame	Perform as a calendar, and a user can select the specific date
ConnectionLinksFrame	ComponentContentFrame	Create/Edit/Delete links from links files

Table 4: CASES Application

V. CONCLUSIONS

A. RESULTS OF THE CASES RESEARCH PROJECT

Automatic software engineering plays a very important role in computer science [BERZ91] [SOMM96]. In this field, a prototyping tool, CAPS, has been designed for automating real-time system development. This research attempted to implement another automatic tool, CASES, to help system analysts and designers carry out their tasks in the software evolution process.

The current version of CASES has verified all of the RH model theories. CASES can perform all of the functions of the RH model including *Automated Version Control* and *Traceability*, which are the most difficult to translate from theory to coding. Without the formalization of software evolution, version control is unclear and the traceability of software evolution is impossible. The RH model provides a good structure of software evolution and CASES provides a practical way of benfitting from this theory.

CASES also has the ability to connect to many application tools and can run on any operating system as long as the system has at least JDK 1.1.7 and Swing 1.0.3 installed. In addition, Microsoft Notepad, Microsoft Word, Microsoft Excel, Netscape, and CAPS should be installed for full performance. Without these application tools, CASES still runs well but connections will be silent.

There probably are defects in CASES since this is the first research version of CASES design to validate the theory of the RH model. Future work on such defects and further improvement of CASES should prove beneficial.

B. RECOMMENDATION

CASES is the first version to prove the practicality of the RH model. Future researchers should continue to build upon and modify CASES to make it an even more effective software evolution tool. The following suggestions should be followed by future researchers to improve the structure, user friendliness, and usefulness of CASES and to avoid unnecessary frustration as well.

CASES 1.1 needs more effort to upgrade the structure to be more reliable. Up until now, no defects have been found, but this build was not targeted to be "Bug-Free". CASES should be upgraded to an operational capability rather than strictly being used for research. CASES 1.1 should have more conditions to restrict a user from inputting inappropriate data. Also, CASES should not allow the user to create an evolution process with the same step as found in other evolution processes. That means each step only occurs in one evolution process. In addition, CASES should not allow CASES users to create input files with invalid steps.

In order to make CASES friendlier, it should have a *Help* menu to explain and give examples of all CASES rules in CasesFrame. CASES should also prevent users from inputting inappropriate data in any text field.

Whenever transferring data from another machine to work on CASES, one should make sure that the receiving machine has the same version of Java Development Kit (JDK), Swing, and Visual Café library as the current machine. Without this, CASES can not open the data from the other machine since the serial identification numbers of these machines are in conflict.

Despite the few areas of opportunity listed above, CASES is a very good software evolution tool. Due to its conformance to and implementation of the RH model, it makes a significant improvement to the automation support for software evolution.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. CASES APPLICATIONS OUTPUT

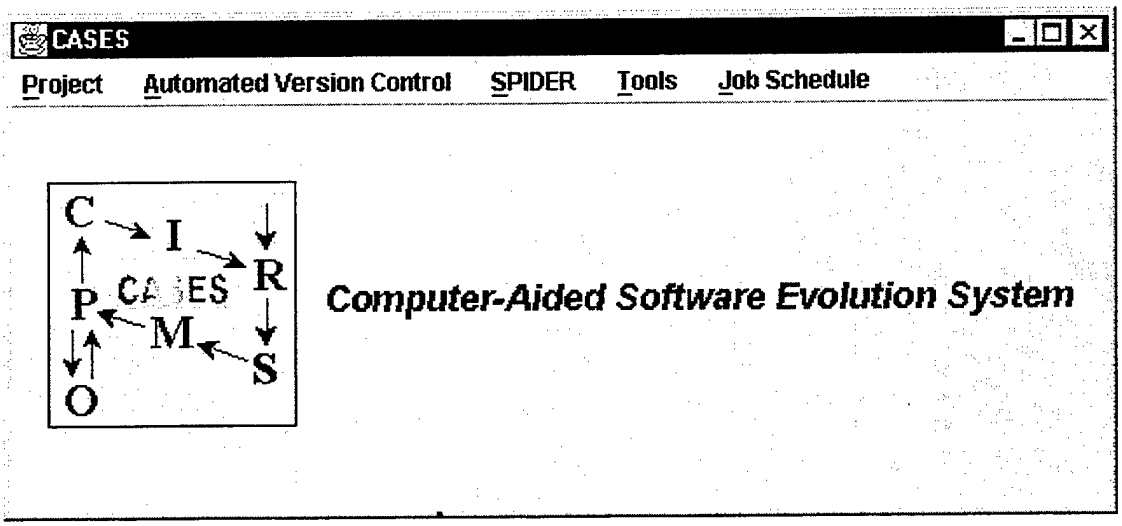


Figure 1: CasesFrame

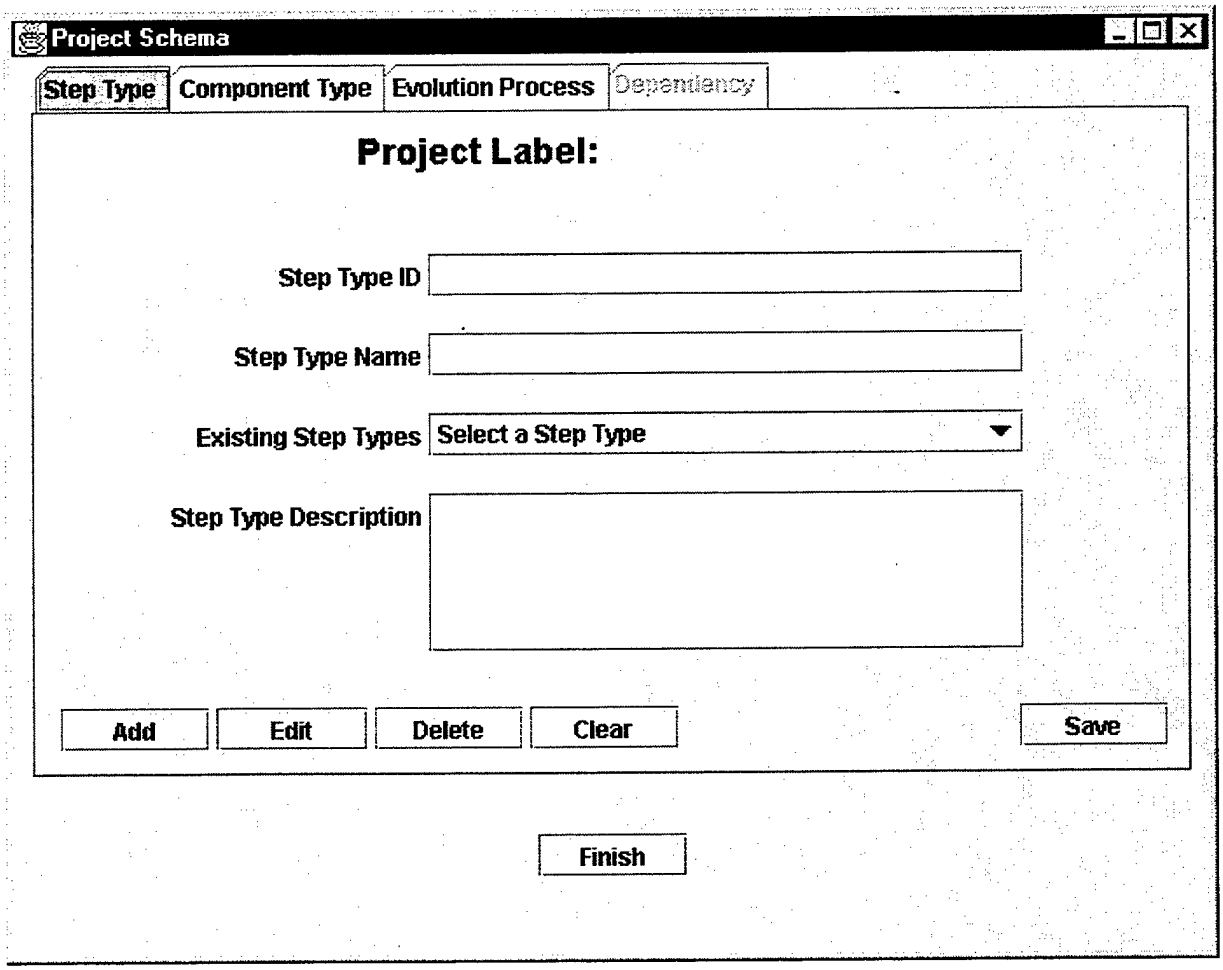


Figure 2: ProjectSchemaFrame

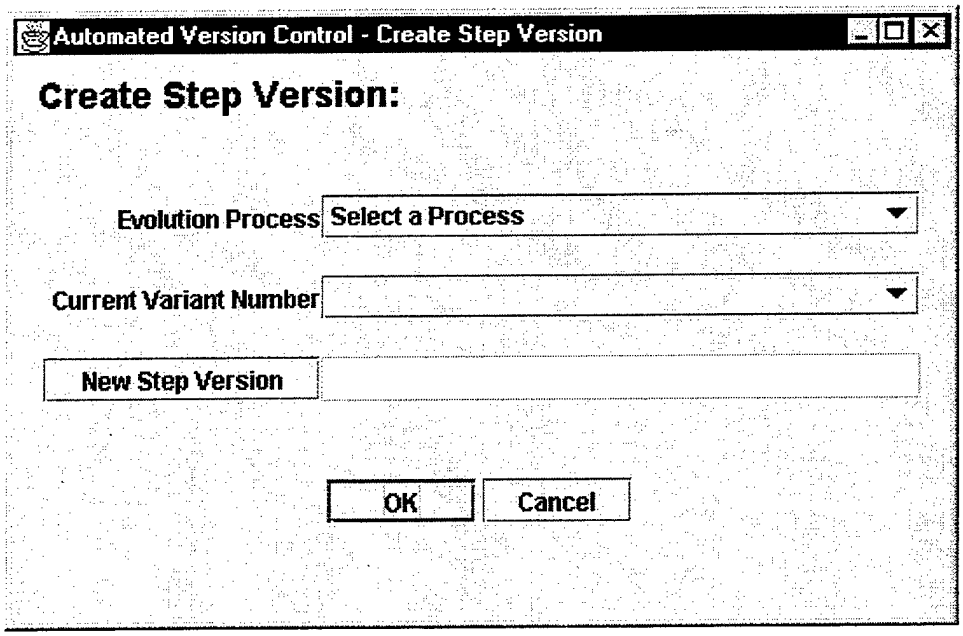


Figure 3: AVCCreateStepFrame

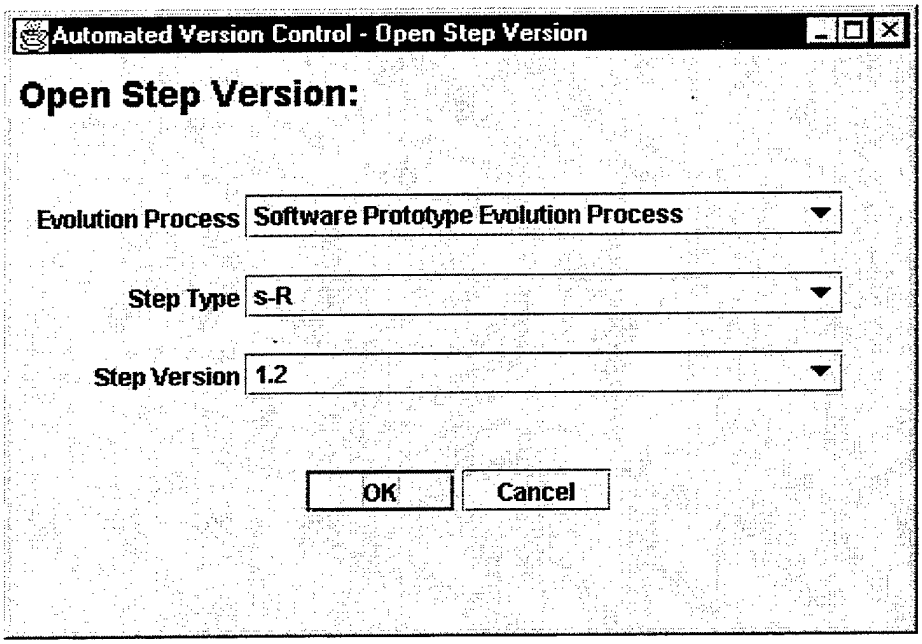


Figure 4: AVCOpenStepFrame

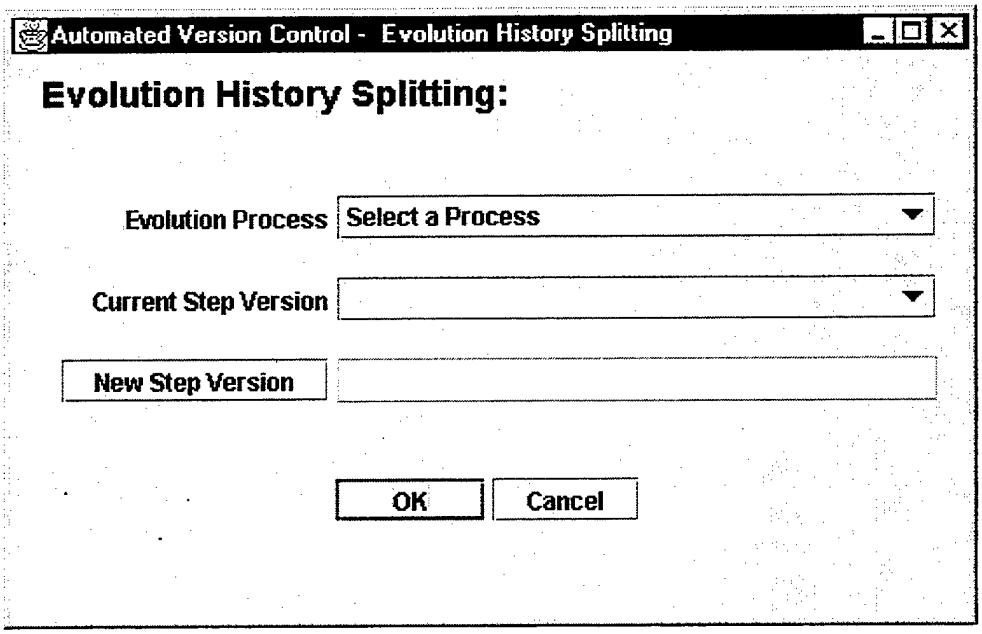


Figure 5: AVCSplittingFrame

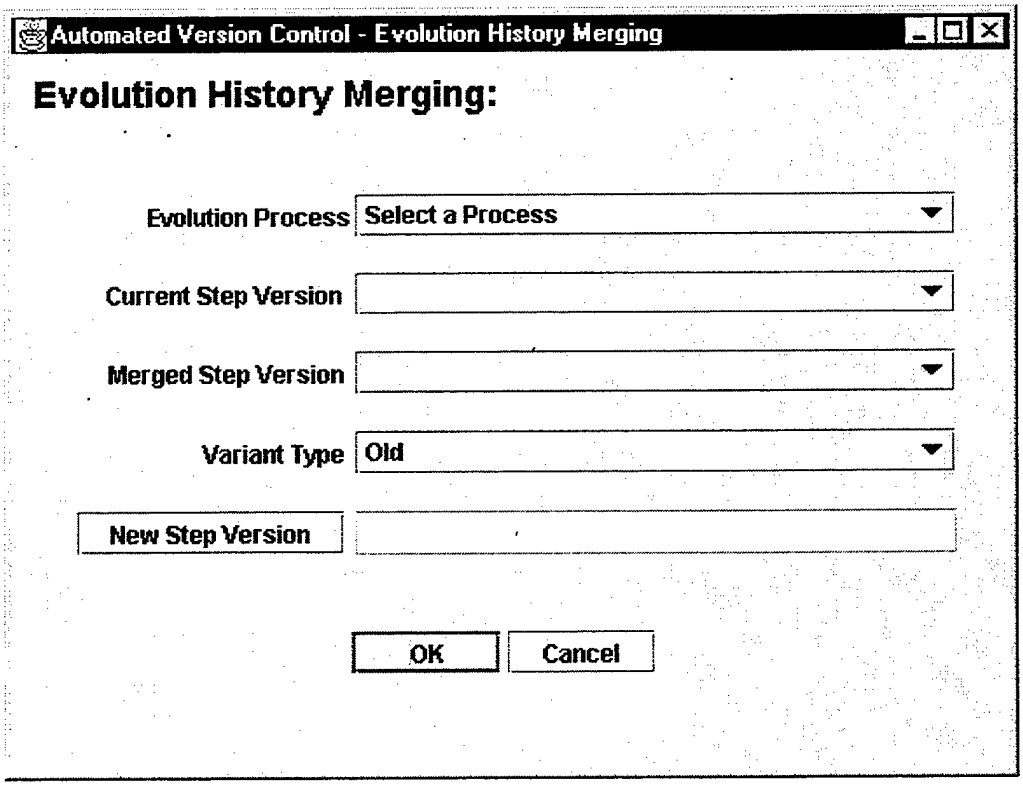


Figure 6: AVCMergingFrame

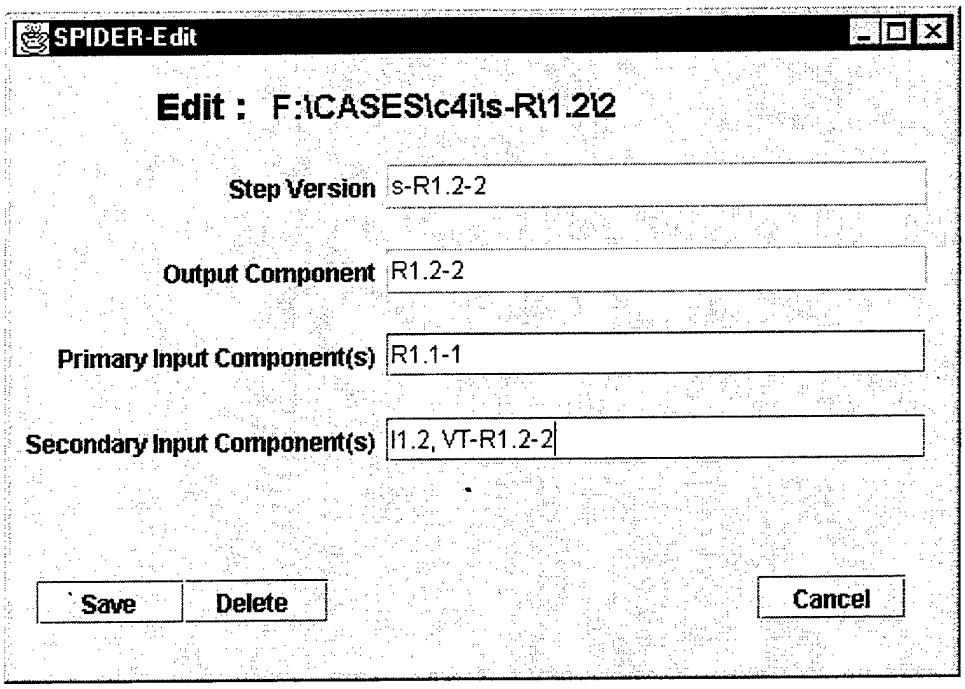


Figure 7: EditDecomposeFrame (Edit)

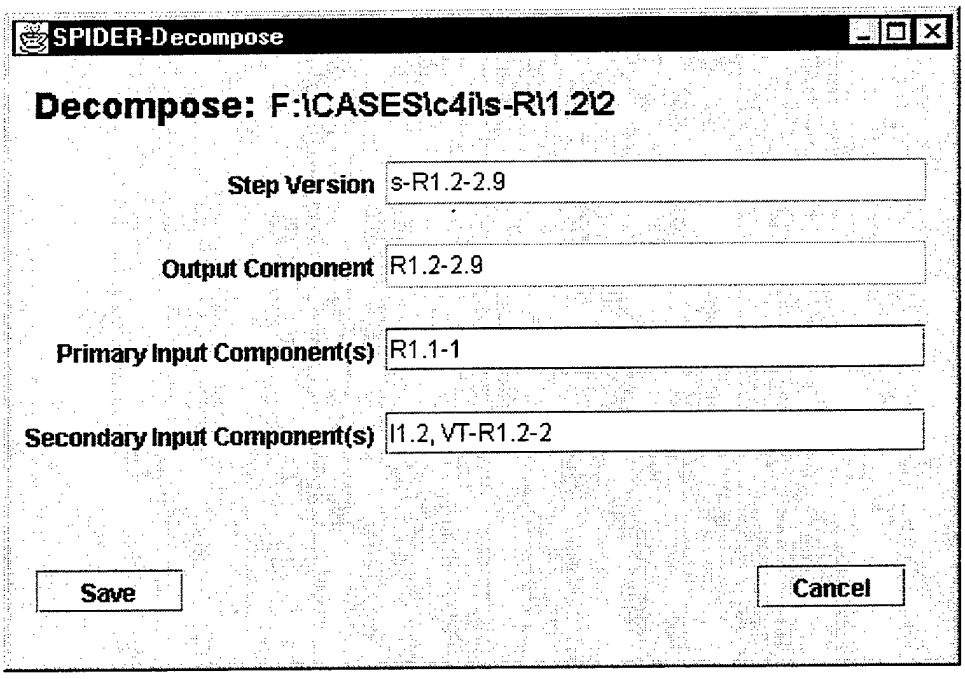


Figure 8: EditDecomposeFrame (Decompose)

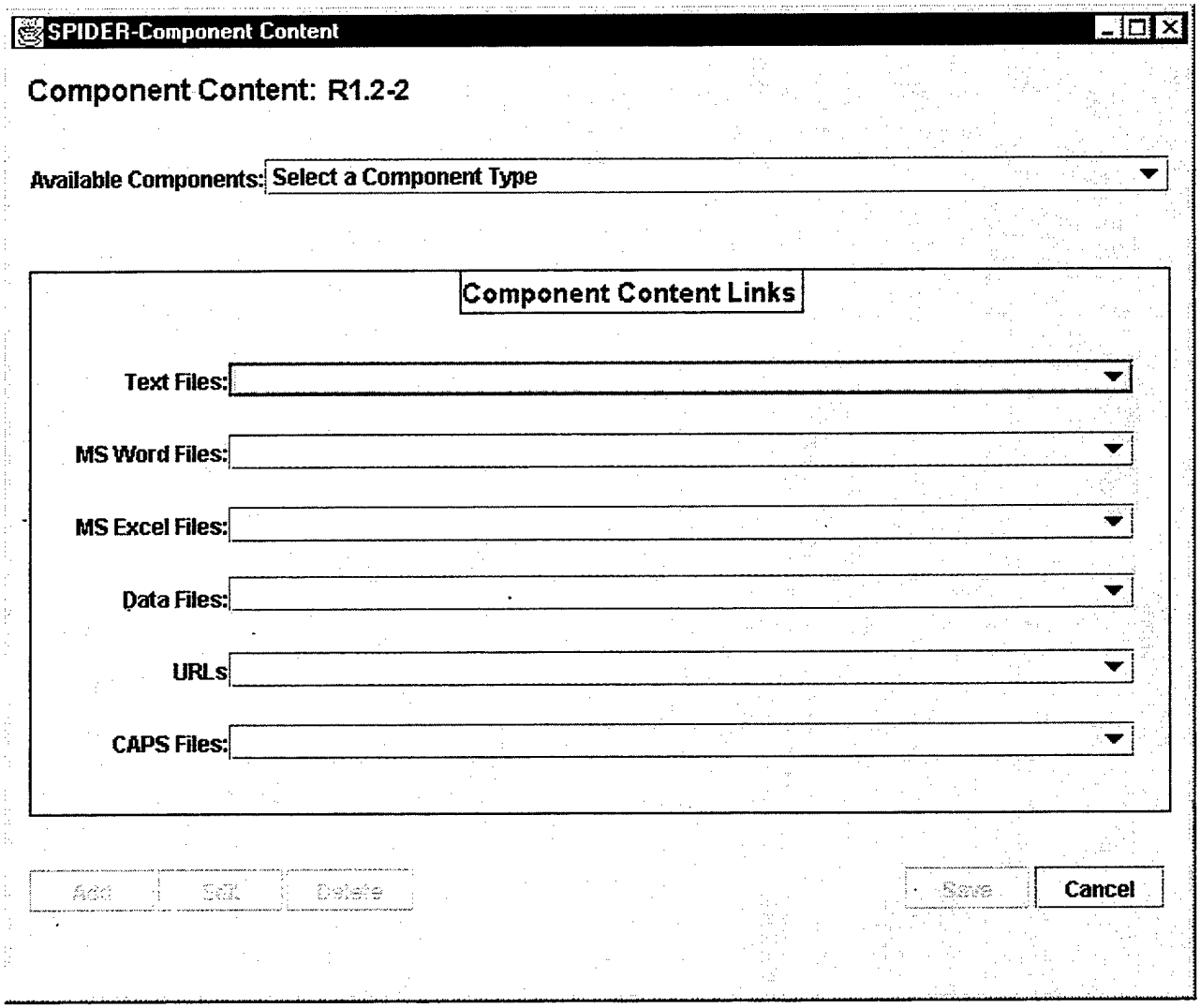


Figure 9: ComponentContentFrame

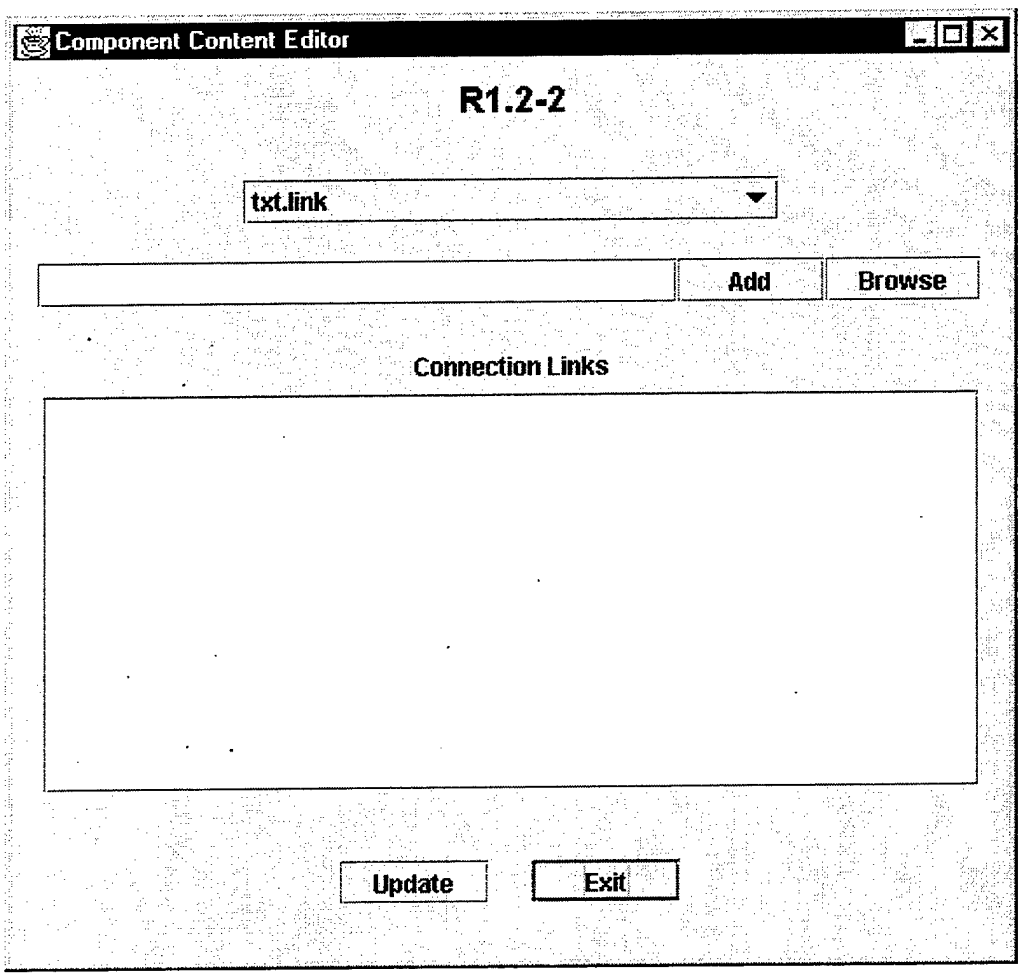


Figure 10: ConnectionLinksFrame

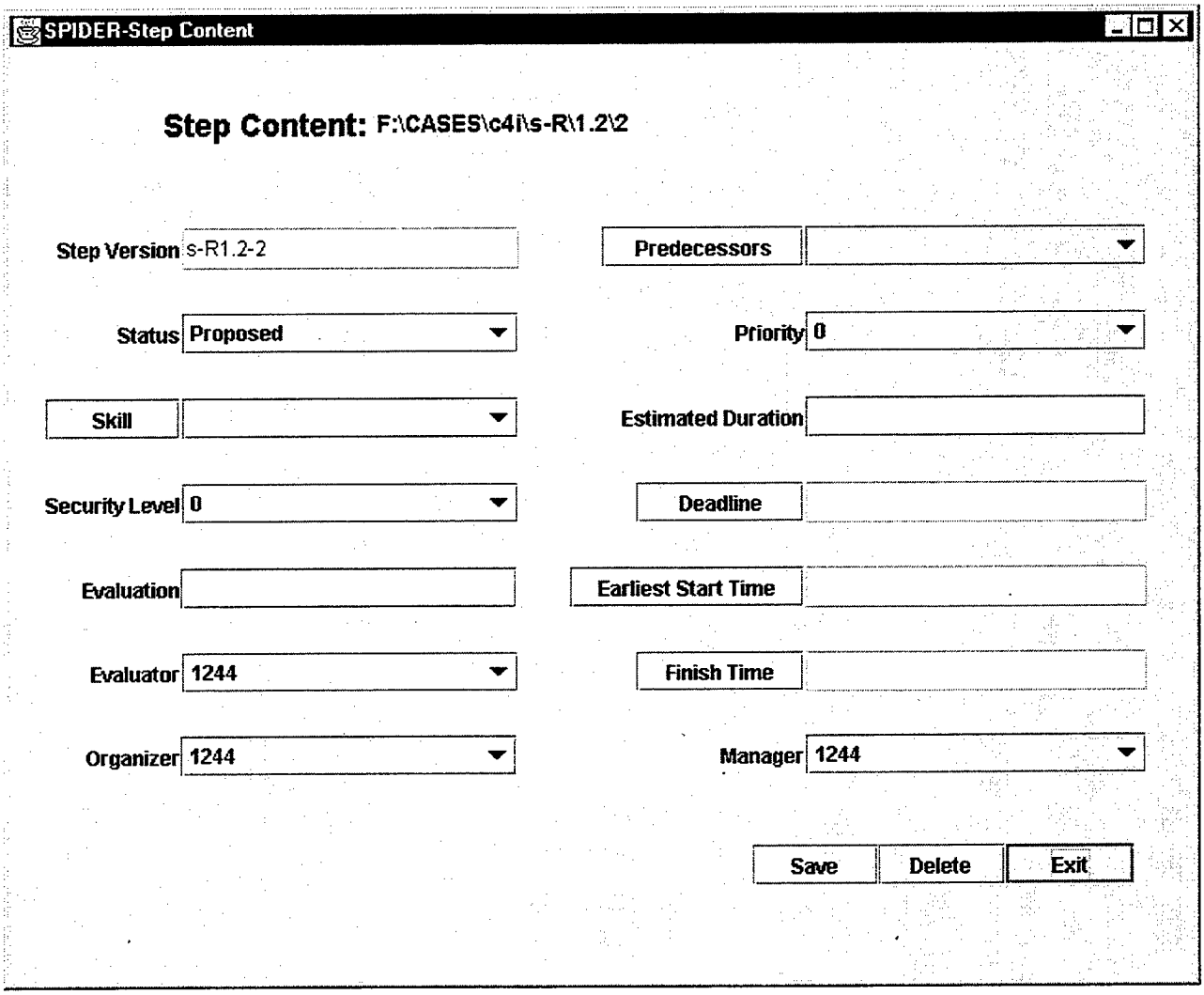


Figure 11: StepContentFrame

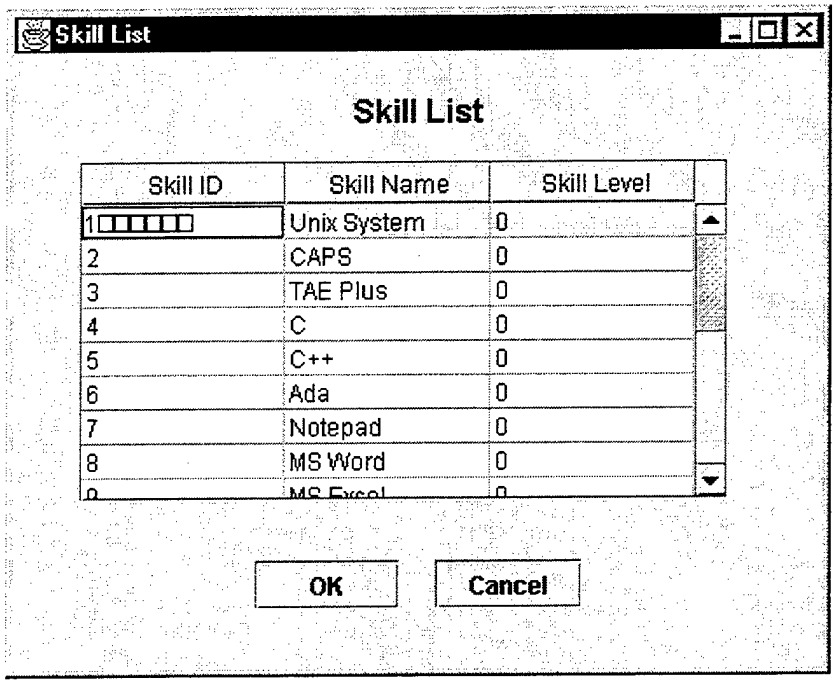


Figure 12: SkillTableFrame

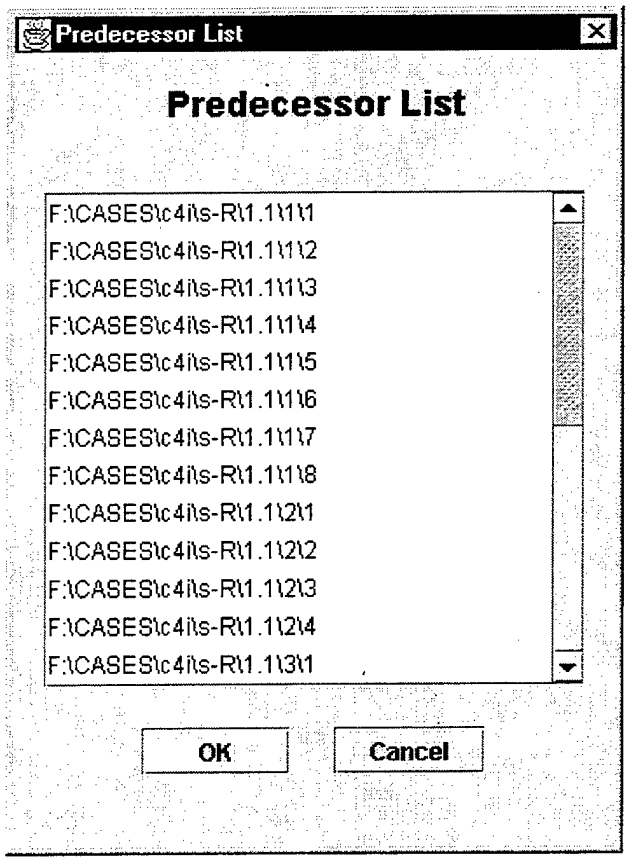


Figure 13: ListDialog (Predecessor List)

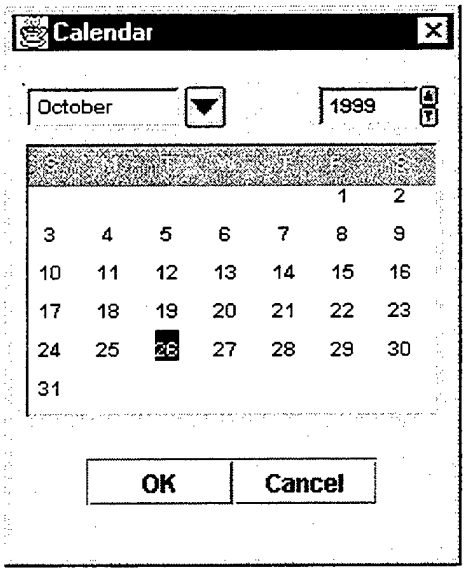


Figure 14: CalendarDialog

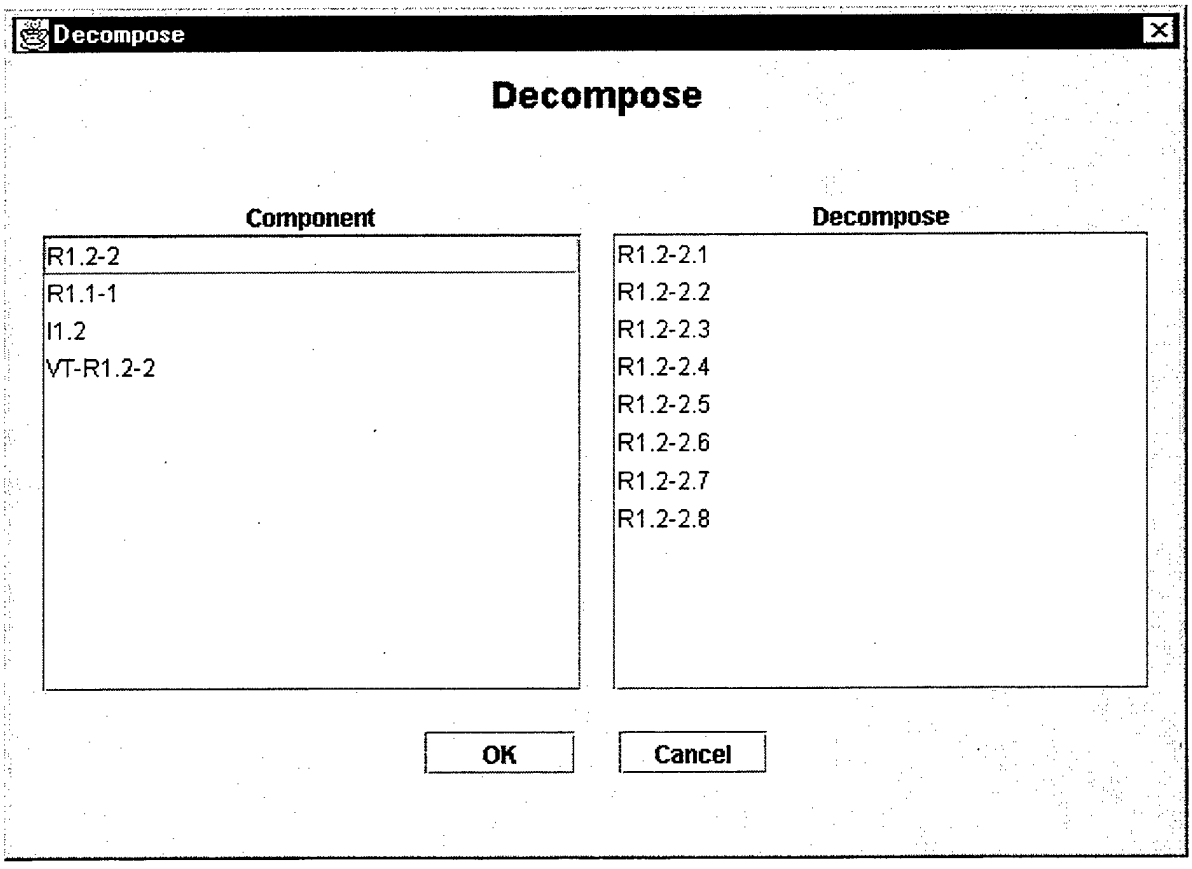


Figure 15: DecomposeListDialog

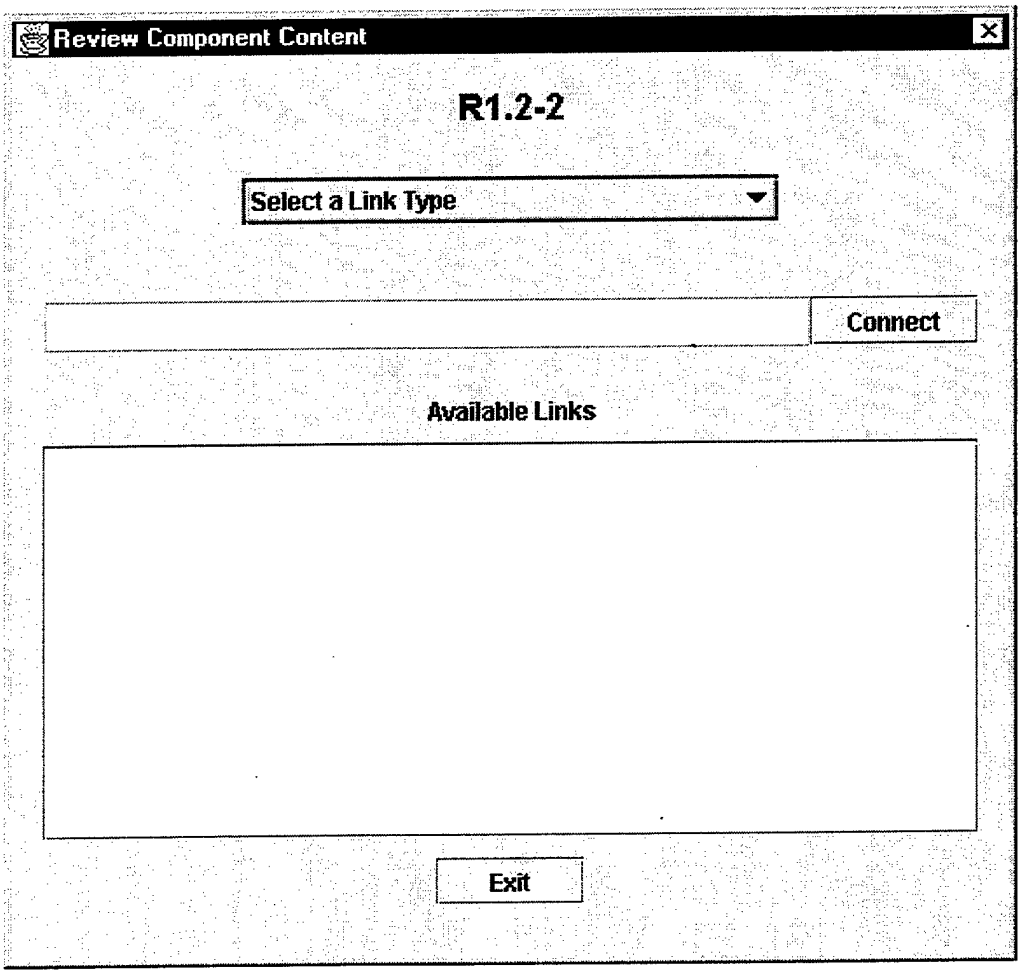


Figure 16: ReviewComponentContentDialog

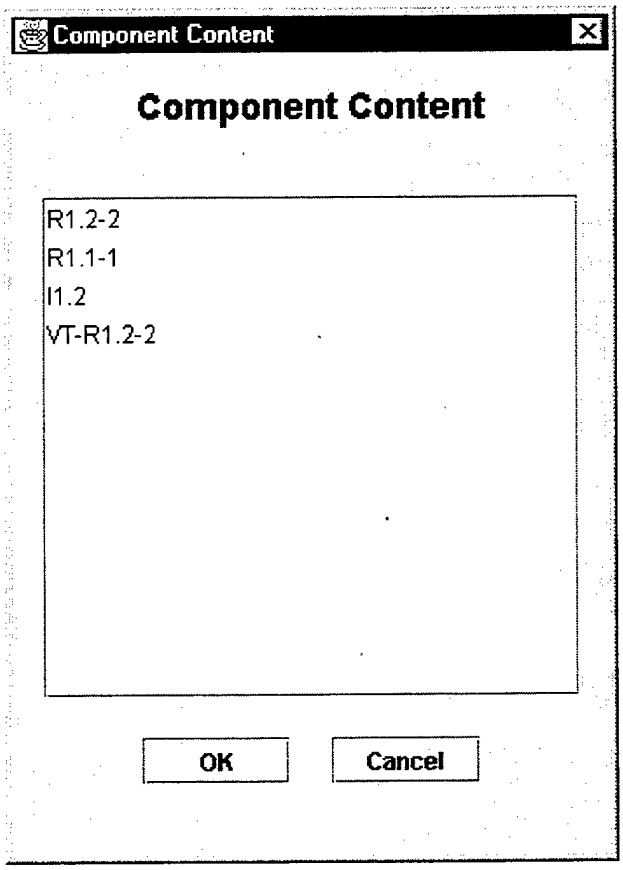


Figure 17: ListDialog (Component Content)

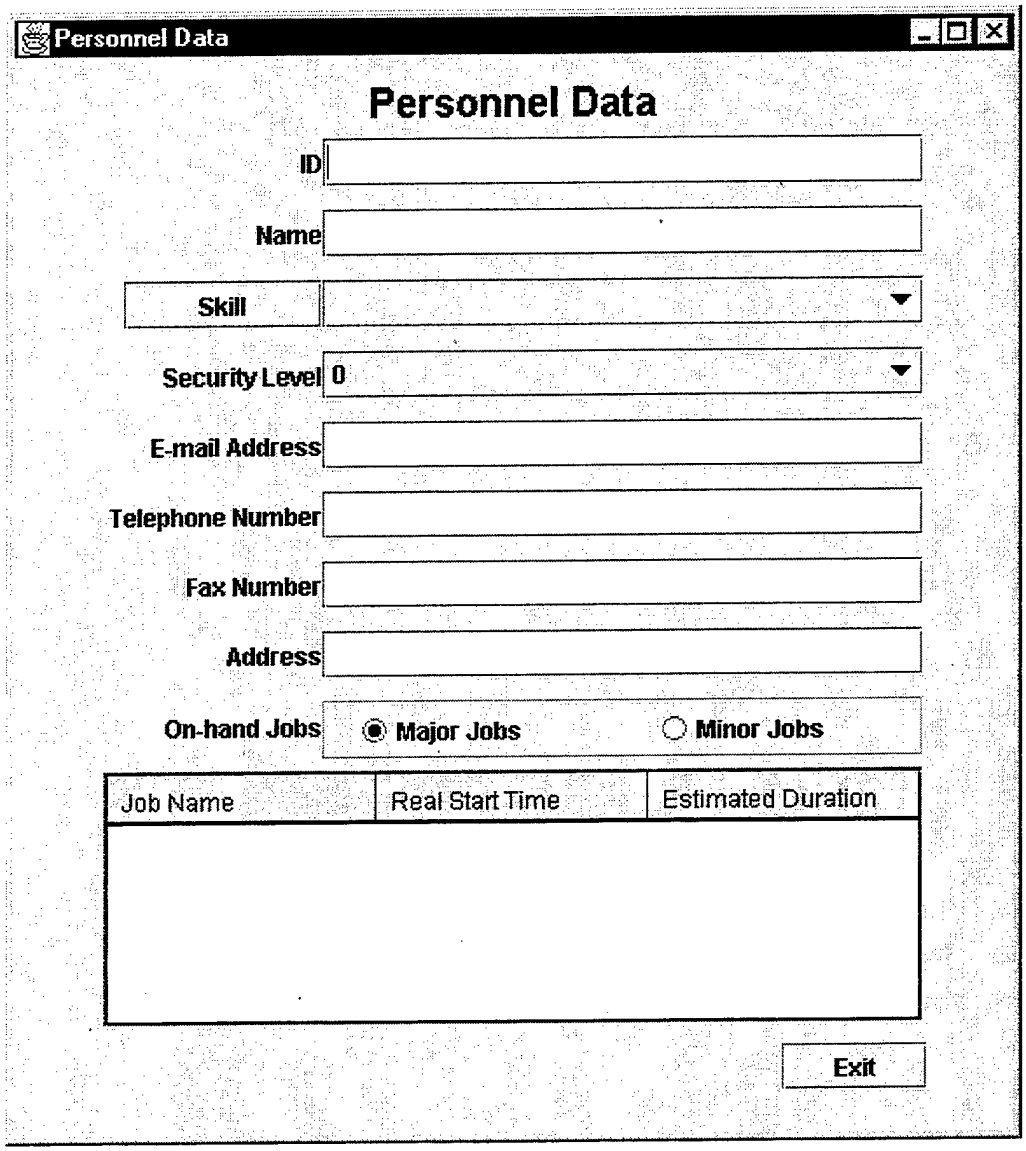


Figure 18: PersonnelFrame (View)

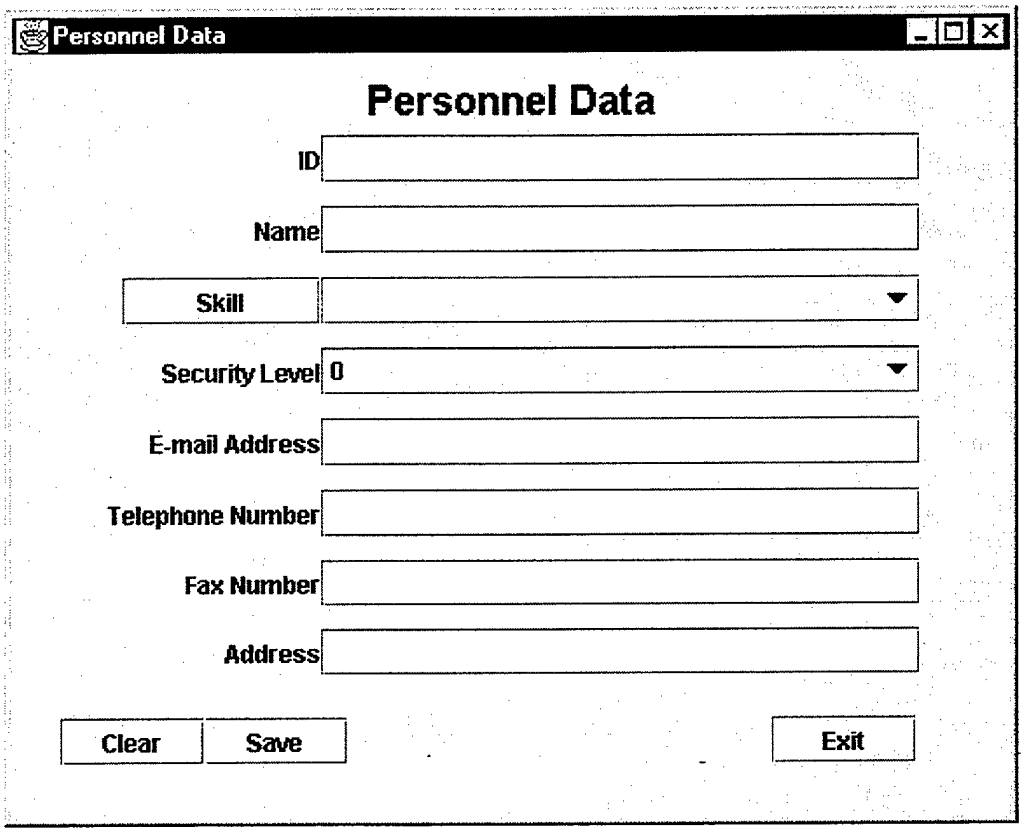


Figure 19: PersonnelFrame (Add/Edit)

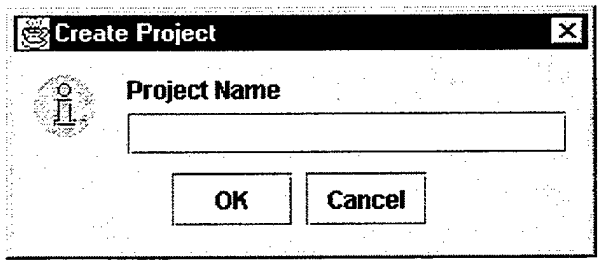


Figure 20: Create Project (JOptionPane, input)

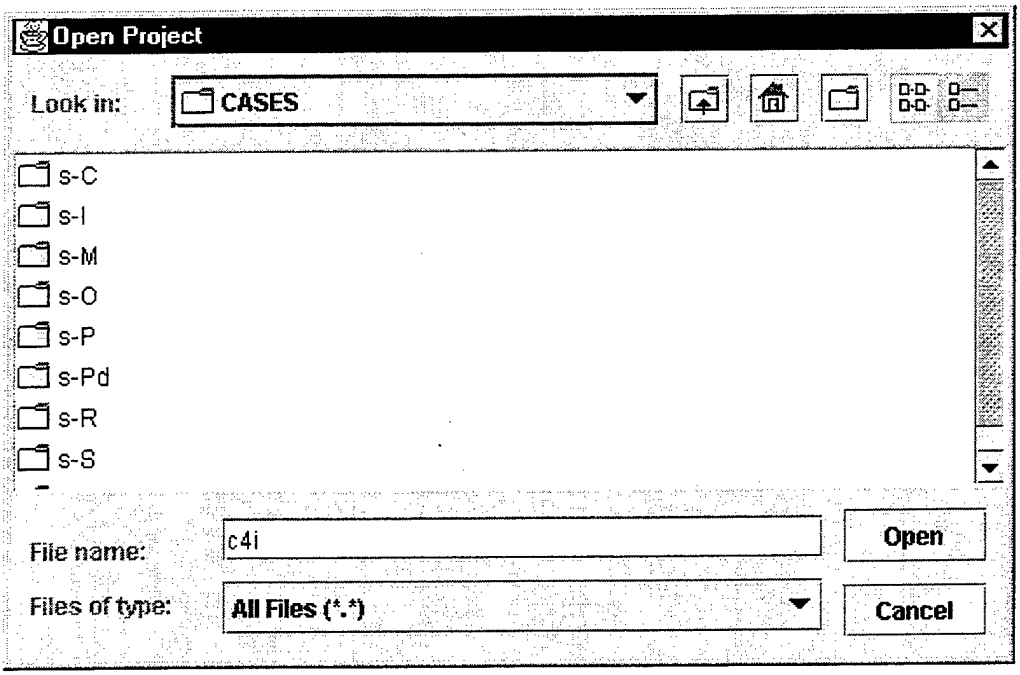


Figure 21: Open Project (JFileChooser)

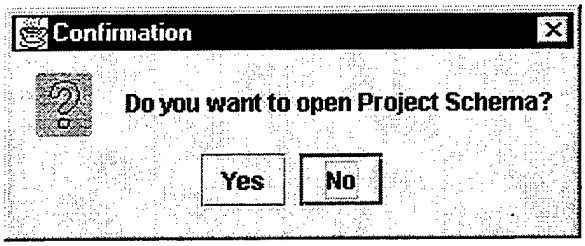


Figure 22: Option Message (JoptionPane-question, confirmation)

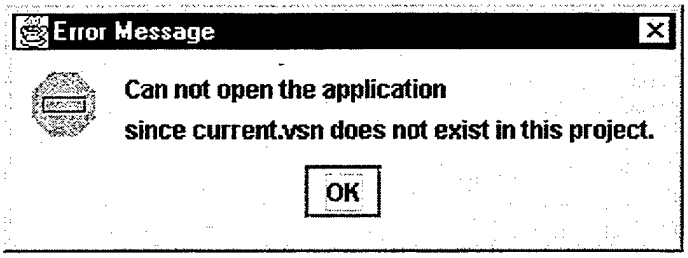


Figure 23: Warning Message (JOptionPane, warning message)

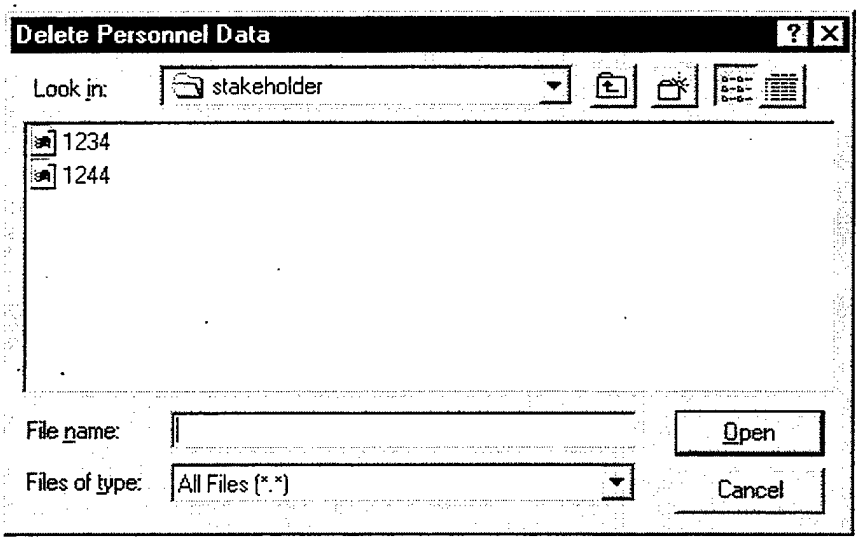


Figure 24: Delete Personnel Date (FileDialog)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SOURCE CODE

```
package Cases;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import com.sun.java.swing.*;
import java.text.*;

////////////////////////////////////////////////////////////////////////
/**
 * Automatic Version Control - Create new version number for all steps
 *    in the current project at the same time.  That means these steps always
 *    have the same number of versions.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_AVC
 */
////////////////////////////////////////////////////////////////////////

public class AVCCreateStepFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_AVC
{
  /**
   * versionVector : stores all version numbers of current project
   */
  public Vector versionVector = new Vector();

  /**
   * pathName : current path name, C:\Cases\projectName
   */
        public String pathName;

  /**
   * EHLHashtable : stores all EHL objects which retrieve from loop.cfg
file
   */
        public Hashtable EHLHashtable = new Hashtable();

  /**
   * depHashtable : stores all Dependency objects which retrieve from
dependency.cfg file
   */
        public Hashtable depHashtable = new Hashtable();


  /**
   * Creating AVCCreateStepFrame
   */
        public AVCCreateStepFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Automated Version Control - Create Step
Version");
                getContentPane().setLayout(null);
                setSize(470,280);
                setVisible(false);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Evolution Process");
                getContentPane().add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(15,70,140,22);
```

```
        currentLabel.setHorizontalAlignment(com.sun.java.swing.SwingC
onstants.RIGHT);
                currentLabel.setText("Current Variant Number");
                getContentPane().add(currentLabel);
                currentLabel.setForeground(java.awt.Color.black);
                currentLabel.setBounds(15,110,140,22);
                OKButton.setText("OK");
                OKButton.setActionCommand("jbutton");
                getContentPane().add(OKButton);
                OKButton.setBounds(158,210,75,22);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("jbutton");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(236,210,75,22);
                newVersionTextField.setEditable(false);
                getContentPane().add(newVersionTextField);

        newVersionTextField.setBackground(java.awt.Color.white);

        newVersionTextField.setForeground(java.awt.Color.black);
                newVersionTextField.setBounds(155,150,300,22);
                getContentPane().add(currentVariantComboBox);
                currentVariantComboBox.setBounds(155,110,300,22);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);

        JLabel1.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel1.setText("Create Step Version: ");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel1.setBounds(10,6,190,25);
        projectLabel.setVerticalAlignment(com.sun.java.swing.SwingCon
stants.BOTTOM);
                projectLabel.setText("Project Label");
                getContentPane().add(projectLabel);
                projectLabel.setForeground(java.awt.Color.black);
                projectLabel.setFont(new Font("Dialog", Font.BOLD,
18));
                projectLabel.setBounds(210,6,200,25);
                getContentPane().add(EHLComboBox);
                EHLComboBox.setBounds(155,70,300,22);
                newStepVersionButton.setText("New Step Version");
                newStepVersionButton.setActionCommand("New Step
Version");
                getContentPane().add(newStepVersionButton);
                newStepVersionButton.setBounds(15,150,140,22);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                SymItem lSymItem = new SymItem();
                currentVariantComboBox.addItemListener(lSymItem);
                EHLComboBox.addItemListener(lSymItem);
                newStepVersionButton.addActionListener(lSymAction);
                //}}
        }

  /**
    * To be called when Create Step Version Menu Item of CasesFrame
    *     receives the event from user.
    * Retrieving Dependency and EHL object from dependency.cfg and
loop.cfg files
```

```
*/
        public AVCCreateStepFrame( String pathName, String dirName ){
    this();
            this.projectLabel.setText( dirName );
            this.pathName = pathName;
    try{
      FileInputStream fileInput = new FileInputStream(
this.pathName+"\\dependency.cfg" );
      ObjectInputStream dep = new ObjectInputStream( fileInput );
      if( dep != null ){
        this.depHashtable = (Hashtable) dep.readObject();
      }
      dep.close();
      fileInput.close();

      fileInput = new FileInputStream( this.pathName+"\\loop.cfg" );
      ObjectInputStream loopIn = new ObjectInputStream( fileInput );
      Vector loopVector = new Vector();
      if( loopIn != null ){
        loopVector = (Vector)loopIn.readObject();
        if( loopVector.size() > 0 ){

          // Setting Evolution Process combobox
          this.setLoopNameComboBox( loopVector );
        }
      }
      loopIn.close();
      fileInput.close();
    }
    catch( IOException e ){ debug("IOException: "+e);
    }
    catch( ClassNotFoundException ex ){
debug("ClassNotFoundException: "+ex);
    }
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new AVCCreateStepFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
```

```
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel currentLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField newVersionTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox currentVariantComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel projectLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox EHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton newStepVersionButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                        else if (object == newStepVersionButton)

                newStepVersionButton_actionPerformed(event);
                        }
                }

  /**
    * Create new directory with new version number for all steps of the
current project
    * @param event, occur when user press OK button
    */
        public void
OKButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          checkPath(this.versionVector);
     setVisible( false );
     dispose();
        }

  /**
    * Exit AVCCreateStepFrame
    * @param event, occur when user press Cancel button
    */
        public void
cancelButton_actionPerformed(java.awt.event.ActionEvent event)
        {
     setVisible( false );
     dispose();
        }

        /**
          * Create new version number and the default version is 1.1
  * @param event, occur when user press New Version button
          */
```

```
public void
newStepVersionButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          if( (EHLComboBox.getItemCount() > 0) &&
(EHLComboBox.getSelectedIndex() > 0) ){
            if( this.currentVariantComboBox.getItemCount() == 0 ){
              newVersionTextField.setText("1.1");
            }
            else{
              createVersionNumber(this.versionVector);
            }
    }
        }

        class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == currentVariantComboBox)

        currentVariantComboBox_itemStateChanged(event);
                        else if (object == EHLComboBox)

        EHLComboBox_itemStateChanged(event);
                }
        }

  /**
    * Allows a user to select all the available Evolution processes in the
combobox
    * @param event, occur when user select Evolution Process
    */
        public void
EHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == ItemEvent.SELECTED ){
        String selectedItem = (String)event.getItem();
        int selectedIndex = this.EHLComboBox.getSelectedIndex();

        if( selectedIndex > 0 ){
          if( this.EHLHashtable.containsKey( selectedItem ) ){
            EHL ehl = (EHL)this.EHLHashtable.get( selectedItem );
            this.versionVector = new Vector();
            this.versionVector =
tokenizeVector((String)ehl.getEHLPath());
            this.setVariantComboBox( this.versionVector) ;
          }
        }
        else if( selectedIndex == 0 ){
          this.currentVariantComboBox.removeAllItems();
          this.newVersionTextField.setText("");
        }
      }
    }

  /**
    * Allows a user to select different variants
    * @param event, occur when user select Variant number
    */
        public void
currentVariantComboBox_itemStateChanged(java.awt.event.ItemEvent
event)
        {
          if( event.getStateChange() == ItemEvent.SELECTED ){
        String currentStep = ((String)event.getItem()).trim();
        newVersionTextField.setText("");
          }
    }

  /**
    * Short cut to print the output
```

```
 * @param string : the output string
 */
      public void debug( String string ){
         System.out.println( string );
      }

/**
 * Tokenizing a string
 * @param string : tokenized string
 * @return v : vector of string without ","
 */
      public Vector tokenizeVector(String string){
  StringTokenizer st = new StringTokenizer( string, "," );
        Vector v = new Vector();
        while( st.hasMoreTokens() ){
          v.addElement( ((String)st.nextToken()).trim() );
        }
  return v;
      }

/**
 * Adding evolution processes into EHLComboBox
 * @param loopVector : vector of evolution processes
 */
      public void setLoopNameComboBox( Vector loopVector ){
  EHLComboBox.removeAllItems();
  EHLComboBox.addItem(PROCESS_TITLE);

  this.EHLHashtable = new Hashtable();

  for( int i=0; i< loopVector.size(); i++ ){
    EHL ehl = (EHL)loopVector.elementAt( i );
    String loopName = ehl.getEHLName();

    //set step Hashtable
    this.EHLHashtable.put( loopName, ehl );
    EHLComboBox.addItem(loopName);
  }
    }

/**
 * Adding variant number of the step into currentVariantComboBox
 * @param theVersionVector : vector of variant number
 */
public void setVariantComboBox( Vector theVersionVector ){
  File aFile = new File(this.pathName,
(String)theVersionVector.elementAt(0));
  if( aFile.isDirectory() ){
    String[] list = aFile.list();

    /* Clean the combo box before update it */
    this.currentVariantComboBox.removeAllItems();
    newVersionTextField.setText("");

    if( list.length > 0 ){
      Vector v = new Vector();
      for( int i=0; i<list.length; i++ ){
        String s = list[i];
        int index = s.indexOf(".");
        String sub1 = (s.substring(0,index)).trim();
        if( !v.contains(sub1) ){
          v.addElement(sub1);
        }
      }
      if( v.size() > 0 ){
        for(int j=0; j<v.size(); j++ ){
          this.currentVariantComboBox.addItem(v.elementAt(j));
        }
      }
    }
  }
}
```

```
/**
 * Create new version number for all steps in the current project
 * @param v : vector of string (e.g., 1.1, 1.2, 2.1, ...) and they
 *          represent all existing version numbers
 */
public void createVersionNumber(Vector v){
        if( v.size() > 0 ){
          try{
            /* To check aFile is existing or not */
            File aFile = new
File(this.pathName,(String)v.elementAt(0));
            if( aFile.isDirectory() ){
              String[] list = aFile.list();
           if( list.length > 0 ){
             int index = list[0].indexOf(".");
             String sub1 =
((String)currentVariantComboBox.getSelectedItem()).trim();
             String sub2 = list[0].substring(index+1);
             int theMax = Integer.parseInt(sub2);
             for( int i=1; i<list.length; i++ ){
               index = list[i].indexOf(".");
               String s = list[i].substring(0, index);
               if( s.equals(sub1)){
                 sub2 = list[i].substring(index+1);
                 int temp = Math.max(theMax,Integer.parseInt(sub2));
                 theMax = temp;
               }
             }
             theMax++;
             String stepVersion =
((String)this.currentVariantComboBox.getSelectedItem()).trim()+"."+theM
ax;
             newVersionTextField.setText(stepVersion);
           }
            }
      }
      catch( Exception e){ }
```

```
        }
  }

  /**
   * Go to each step (e.g., s-I, s-C, s-R, ...) and create new directory with
   *  the name is new version number
   * @param v : vector of step names
   */
  public void checkPath(Vector v){
    String currentVersion = this.newVersionTextField.getText();
     if( v.size() > 0 ){
       File myFile = new File(this.pathName);
       if( myFile.exists()) {
         String[] myList = myFile.list();
         for(int m=0; m<myList.length; m++ ){
           File f = new File(myFile, (String)myList[m]);
           if( f.isDirectory() ){
             if( v.contains(f.getName()) ){
               File theFile = new File(f,currentVersion);
               theFile.mkdir();
               if( !theFile.isDirectory() ){
                 theFile.mkdir();
               }
               if( theFile.isDirectory() ){
                 Dependency dep = (Dependency)
this.depHashtable.get( f.getName() );
                 createFiles(dep, theFile);
               }
             }
           }
         }
       }
     }
  }

  /**
   * Create Component Content directory and link files inside it.
```

```
    * @param dep : Dependency object of theFile and get input.p and
input.s files
    * @param theFile : current version directory
    */
   public void createFiles( Dependency dep, File theFile){
      if( dep !=  null ){
         try{
            //Create Component Content subdirectory in thte new step
            //and include txt.link, word.link, excel.link, data.link, url.link, and
caps.link
            File compContent = new File(theFile,"Component Content");
            compContent.mkdir();
            if( compContent.isDirectory() ){
               for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
                  File newFile = new File(compContent,
LINK_FILE_NAMES[i]);
                  FileWriter fileWriter = new FileWriter( newFile );
                  BufferedWriter bw = new BufferedWriter( fileWriter );
                  bw.flush();
                  bw.close();
                  fileWriter.close();
               }
            }

            FileOutputStream fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.p");
            DataOutputStream depPrimary = new DataOutputStream(
fileOutput);
            if( depPrimary != null ){
               depPrimary.writeBytes(dep.getPrimaryInput());
            }
            depPrimary.flush();
            depPrimary.close();
            fileOutput.close();

            fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.s");
            DataOutputStream depSecondary = new DataOutputStream(
fileOutput);
            if( depSecondary != null ){
               depSecondary.writeBytes(dep.getSecondaryInput());
            }
            depSecondary.flush();
            depSecondary.close();
            fileOutput.close();
         }
         catch( IOException e ){ debug("dep_IOException: "+e);
         }
      }
   }
}
```



```
package Cases;

import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import java.text.*;

////////////////////////////////////////////////////////////////////////
/**
 * Automatic Version Control - Evolution History Merging : to merge two
existing
 *    versions, and create the new version.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_AVC
 */
////////////////////////////////////////////////////////////////////////
```

```
public class AVCMergingFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_AVC
{
  /**
   * pathName : current path name, e.g. C:\Cases\projectName,
D:\Cases\projectName, ...
   */
        public String pathName;

  /**
   * versionVector : stores all version numbers of current project
   */
  public Vector versionVector = new Vector();

  /**
   * EHLHashtable : stores all EHL objects which retrieve from loop.cfg
file
   */
        public Hashtable EHLHashtable = new Hashtable();

 /**
   * depHashtable : stores all Dependency objects which retrieve from
dependency.cfg file
   */
        public Hashtable depHashtable = new Hashtable();

 /**
   * Creating AVCMergingFrame
   */
        public AVCMergingFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Automated Version Control - Evolution History
Merging");
                getContentPane().setLayout(null);
                setSize(500,360);
                setVisible(false);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Evolution Process");
                getContentPane().add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(26,70,145,22);

        currentStepLabel.setHorizontalAlignment(com.sun.java.swing.Swi
ngConstants.RIGHT);
                currentStepLabel.setText("Current Step Version");
                getContentPane().add(currentStepLabel);
                currentStepLabel.setForeground(java.awt.Color.black);
                currentStepLabel.setBounds(23,110,144,22);

        JLabel11.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel11.setText("Variant Type");
                getContentPane().add(JLabel11);
                JLabel11.setForeground(java.awt.Color.black);
                JLabel11.setBounds(23,190,145,22);
                OKButton.setText("OK");
                OKButton.setActionCommand("jbutton");
                getContentPane().add(OKButton);
                OKButton.setBounds(171,290,75,22);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("jbutton");
                getContentPane().add(cancelButton);
```

```
                cancelButton.setBounds(249,290,75,22);
                getContentPane().add(currentStepComboBox);
                currentStepComboBox.setBounds(173,110,300,22);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);

        JLabel1.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel1.setText("Evolution History Merging: ");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel1.setBounds(0,6,250,25);

        projectLabel.setVerticalAlignment(com.sun.java.swing.SwingCon
stants.BOTTOM);
                projectLabel.setText("Project Label");
                getContentPane().add(projectLabel);
                projectLabel.setForeground(java.awt.Color.black);
                projectLabel.setFont(new Font("Dialog", Font.BOLD,
18));
                projectLabel.setBounds(250,6,250,25);
                getContentPane().add(EHLComboBox);
                EHLComboBox.setBounds(173,70,300,22);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel2.setText("Merged Step Version");
                getContentPane().add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2.setBounds(23,150,145,22);
                getContentPane().add(mergedStepComboBox);
                mergedStepComboBox.setBounds(173,150,300,22);
                newVersionTextField.setEditable(false);
                getContentPane().add(newVersionTextField);
                newVersionTextField.setBackground(java.awt.Color.white);

                newVersionTextField.setForeground(java.awt.Color.black);
                        newVersionTextField.setBounds(173,230,300,22);
                        getContentPane().add(variantComboBox);
                        variantComboBox.setBounds(173,190,300,22);
                        newStepVersionButton.setText("New Step Version");
                        newStepVersionButton.setActionCommand("New Step
Version");
                        getContentPane().add(newStepVersionButton);
                        newStepVersionButton.setBounds(33,230,135,22);
                        //}}

                        //{{INIT_MENUS
                        //}}

                        //{{REGISTER_LISTENERS
                        SymAction lSymAction = new SymAction();
                        OKButton.addActionListener(lSymAction);
                        cancelButton.addActionListener(lSymAction);
                        SymItem lSymItem = new SymItem();
                        EHLComboBox.addItemListener(lSymItem);
                        newStepVersionButton.addActionListener(lSymAction);
                        currentStepComboBox.addItemListener(lSymItem);
                        mergedStepComboBox.addItemListener(lSymItem);
                        variantComboBox.addItemListener(lSymItem);
                        //}}

    /* Adding item to variantComboBox */
    for( int i=0; i<VARIANT_TYPES.length; i++ ){
            variantComboBox.addItem(VARIANT_TYPES[i]);
        }
        variantComboBox.setSelectedIndex( 0 );
    }

    /**
```

```
    * To be called when Evolution History Merging Menu Item of
CasesFrame
    *     receives the event from user.
    * Retrieving Dependency and EHL object from dependency.cfg and
loop.cfg files
    */
   public AVCMergingFrame( String pathName, String dirName ){
      this();
            this.projectLabel.setText( dirName );
            this.pathName = pathName;

      try{
         FileInputStream fileInput = new FileInputStream(
this.pathName+"\\dependency.cfg" );
         ObjectInputStream dep = new ObjectInputStream( fileInput );
         if( dep != null ){
            this.depHashtable = (Hashtable) dep.readObject();
         }
         dep.close();
         fileInput.close();
      }
      catch( IOException e ){
         debug("IOException_Dep: "+e);
      }
      catch( ClassNotFoundException ex ){
         debug("ClassNotFoundException_Dep: "+ex);
      }

      try{
         FileInputStream fileInput = new FileInputStream(
this.pathName+"\\loop.cfg" );
         ObjectInputStream loopIn = new ObjectInputStream( fileInput );
         if( loopIn != null ){
            Vector loopVector = new Vector();
            loopVector = (Vector)loopIn.readObject();
            if( loopVector.size() > 0){
               this.setLoopNameComboBox( loopVector );
            }
         }
         loopIn.close();
         fileInput.close();
      }
      catch( IOException e ){
         debug("IOException_loop: "+e);
      }
      catch( ClassNotFoundException ex ){
         debug("ClassNotFoundException_loop: "+ex);
      }
   }

      public void setVisible(boolean b)
      {
            if (b)
                  setLocation(50, 50);
            super.setVisible(b);
      }

      static public void main(String args[])
      {
            (new AVCMergingFrame()).setVisible(true);
      }

      public void addNotify()
      {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                  return;
            frameSizeAdjusted = true;
```

```
                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel currentStepLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel11 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JComboBox currentStepComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel projectLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox EHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox mergedStepComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextField newVersionTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox variantComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton newStepVersionButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                        else if (object == newStepVersionButton)

        newStepVersionButton_actionPerformed(event);
                }
        }

  /**
   * Create new directory with new version number for all steps of the
current project
   * @param event, occur when user press OK button
   */
  public void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
```

```
        {
    checkPath(this.versionVector);
    setVisible( false );
    dispose();
        }

  /**
   * Exit AVCMergingFrame
   * @param event, occur when user press Cancel button
   */
        public void
cancelButton_actionPerformed(java.awt.event.ActionEvent event)
        {
    setVisible( false );
    dispose();
        }

        /**
         * Create new version number and the default version is 1.1
   * @param event, occur when user press New Version button
         */
  public void
newStepVersionButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          if( (EHLComboBox.getItemCount() > 0) &&
(EHLComboBox.getSelectedIndex() > 0) ){
            if( this.currentStepComboBox.getItemCount()>0 &&
              this.mergedStepComboBox.getItemCount() > 0 ){
            createVersionNumber(variantComboBox.getSelectedIndex());
            }
            else{
              newVersionTextField.setText("1.1");
            }
    }
        }
```

```
        class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == EHLComboBox)

        EHLComboBox_itemStateChanged(event);
                        else if (object == currentStepComboBox)

        currentStepComboBox_itemStateChanged(event);
                else if (object == mergedStepComboBox)

        currentStepComboBox_itemStateChanged(event);
                        else if (object == variantComboBox)

        currentStepComboBox_itemStateChanged(event);
                }
        }

  /**
   * Allows a user to select all the available Evolution processes in the
combobox
   * @param event, occur when user select Evolution Process
   */
  public void
EHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
                if( event.getStateChange() == ItemEvent.SELECTED ){
        String selectedItem = (String)event.getItem();
        int selectedIndex = this.EHLComboBox.getSelectedIndex();

        this.currentStepComboBox.removeAllItems();
        this.mergedStepComboBox.removeAllItems();
        newVersionTextField.setText("");
```

```
        if( selectedIndex > 0 ){
          if( this.EHLHashtable.containsKey( selectedItem ) ){
            EHL ehl = (EHL)this.EHLHashtable.get( selectedItem );
            this.versionVector = new Vector();
            this.versionVector =
tokenizeVector((String)ehl.getEHLPath());
            this.setVersionComboBoxes( this.versionVector) ;
          }
        }
      }
    }

    /**
     * Allows a user to select all the available steps in the combobox
     * @param event, occur when user select currentStepComboBox
     */
          public void
currentStepComboBox_itemStateChanged(java.awt.event.ItemEvent event)
          {
      if( event.getStateChange() == ItemEvent.SELECTED ){
        newVersionTextField.setText("");
      }
          }

    /**
     * Short cut to print the output
     * @param string : the output string
     */
          public void debug( String string ){
            System.out.println( string );
          }

    /**
     * Adding evolution processes into EHLComboBox
     * @param loopVector : vector of evolution processes
     */
          public void setLoopNameComboBox( Vector loopVector ){
      EHLComboBox.removeAllItems();
      EHLComboBox.addItem(PROCESS_TITLE);

      this.EHLHashtable = new Hashtable();

      for( int i=0; i< loopVector.size(); i++ ){
        EHL ehl = (EHL)loopVector.elementAt( i );
        String loopName = ehl.getEHLName();

        //set step Hashtable
        this.EHLHashtable.put( loopName, ehl );

        this.EHLComboBox.addItem(loopName);
      }
          }

    /**
     * Adding version numbers to currentStepComboBox and
mergedStepComboBox
     * @param versionVector : vector of version numbers
     */
    public void setVersionComboBoxes( Vector versionVector ){

      File aFile = new File(this.pathName,
(String)versionVector.elementAt(0));
      if( aFile.isDirectory() ){
        String[] list = aFile.list();

        if( list.length > 0 ){
          for( int i=0; i<list.length; i++ ){
            this.currentStepComboBox.addItem(((String)list[i]).trim());
            this.mergedStepComboBox.addItem(((String)list[i]).trim());
          }
        }
      }
    }
```

```
/**
 * Tokenizing a string
 * @param string : tokenized string
 * @return v : vector of string without ","
 */
      public Vector tokenizeVector(String string){
  StringTokenizer st = new StringTokenizer( string, "," );
        Vector v = new Vector();
        while( st.hasMoreTokens() ){
          v.addElement( ((String)st.nextToken()).trim() );
        }
  return v;
      }

/**
 * Create new version number for all steps in the current project
 * @param typeIndex = 0 ==> variant is old, new version =
currentVariant.(highestVersion+1);
 *              = 1 ==> variant is new, new version =
(highestVariant+1).(highestVersion+1);
 */
 public void createVersionNumber(int typeIndex){
        String current =
((String)currentStepComboBox.getSelectedItem()).trim();
        String merged =
((String)mergedStepComboBox.getSelectedItem()).trim();

        if( current.equals(merged) ){
      JOptionPane.showMessageDialog(this, " Current Step Version must
be different from Merged Step Version!",
                    "Error Message",
JOptionPane.ERROR_MESSAGE);
      return;
        }
        else{
      try{
        int index1 = current.indexOf(".");
        int index2 = merged.indexOf(".");
        String sub1 = current.substring(index1+1);
        String sub2 = merged.substring(index2+1);
        int theMax2 =
Math.max(Integer.parseInt(sub1),Integer.parseInt(sub2));
        theMax2++;
        String mergedVersion = null;

        if( typeIndex == 0 ){
          mergedVersion = current.substring(0,index1)+"."+theMax2;
        }
        else if( typeIndex == 1 ){
          index1 = current.indexOf(".");
          index2 = merged.indexOf(".");
          sub1 = current.substring(0, index1);
          sub2 = merged.substring(0, index2);
          int theMax1 =
Math.max(Integer.parseInt(sub1),Integer.parseInt(sub2));
          theMax1++;
          mergedVersion = theMax1+"."+theMax2;
        }

        for( int i=0; i<currentStepComboBox.getItemCount(); i++ ){
          String s = (String)currentStepComboBox.getItemAt(i);
          if( s.equals(mergedVersion) ){
            JOptionPane.showMessageDialog(this, mergedVersion+"
version already exists in this project!",
                    "Error Message",
JOptionPane.ERROR_MESSAGE);
            return;
          }
          else if( i==currentStepComboBox.getItemCount()-1){
            newVersionTextField.setText(mergedVersion);
          }
        }
      }
      catch( Exception e){}
```

```
    }
  }

  /**
   * Go to each step (e.g., s-I, s-C, s-R, ...) and create new directory with
   *  the name is new version number
   * @param v : vector of step names
   */
  public void checkPath(Vector v){
    String mergedVersion = this.newVersionTextField.getText();
    if( v.size() > 0 ){
      File myFile = new File(this.pathName);
      if( myFile.exists()) {
        String[] myList = myFile.list();
        for(int m=0; m<myList.length; m++ ){
          File f = new File(myFile, (String)myList[m]);
          if( f.isDirectory() ){
            if( v.contains(f.getName()) ){
              File theFile = new File(f,mergedVersion);
              theFile.mkdir();
              if( !theFile.isDirectory() ){
                theFile.mkdir();
              }
              if( theFile.isDirectory() ){
                Dependency dep = (Dependency)
this.depHashtable.get( f.getName() );
                  createFiles(dep, theFile);
              }
            }
          }
        }
      }
    }
  }

  /**
   * Create Component Content directory and link files inside it.
   * @param dep : Dependency object of theFile and get input.p and
input.s files
   * @param theFile : current version directory
   */
  public void createFiles( Dependency dep, File theFile){
    if( dep !=  null ){
      try{
        //Create Component Content subdirectory in thte new step
        //and include txt.link, data.link, url.link, and caps.link
        File compContent = new File(theFile,"Component Content");
        compContent.mkdir();
        if( compContent.isDirectory() ){
          for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
            File newFile = new File(compContent,
LINK_FILE_NAMES[i]);
            FileWriter fileWriter = new FileWriter( newFile );
            BufferedWriter bw = new BufferedWriter( fileWriter );
            bw.flush();
            bw.close();
            fileWriter.close();
          }
        }

        FileOutputStream fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.p");
        DataOutputStream depPrimary = new DataOutputStream(
fileOutput);
        if( depPrimary != null ){
          depPrimary.writeBytes(dep.getPrimaryInput());
        }
        depPrimary.flush();
        depPrimary.close();
        fileOutput.close();

        fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.s");
```

```
            DataOutputStream depSecondary = new DataOutputStream(
fileOutput);
            if( depSecondary != null ){
               depSecondary.writeBytes(dep.getSecondaryInput());
            }
            depSecondary.flush();
            depSecondary.close();
            fileOutput.close();
         }
         catch( IOException e ){
            debug("dep_IOException: "+e);
         }
      }
   }
}
```



```
package Cases;

import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;

//////////////////////////////////////////////////////////////////////////////
/**
 * Automatic Version Control - Open existing version number for all steps
 *    in the current project.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_AVCOpenStep
 */
//////////////////////////////////////////////////////////////////////////////
public class AVCOpenStepFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_AVCOpenStep
{
   /**
    * pathName : current path name, e.g. C:\Cases\projectName,
D:\Cases\projectName, ...
    */
   public String pathName;

   /**
    * EHLHashtable : stores all EHL objects which retrieve from loop.cfg
file
    */
         public Hashtable EHLHashtable = new Hashtable();

         /**
          * oldVersionControl : VersionControl object, keeping the content
of current.cfg file
          */
         protected VersionControl oldVersionControl = null;


   /**
    * Creating AVCOpenStepFrame
    */
         public AVCOpenStepFrame()
         {
                  // This code is automatically generated by Visual Cafe
when you add
                  // components to the visual environment. It instantiates
and initializes
                  // the components. To modify the code, only use code
syntax that matches
                  // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                  // parse your Java file into its visual environment.
                  //{{INIT_CONTROLS
                  setTitle("Automated Version Control - Open Step
Version");
```

```
                getContentPane().setLayout(null);
                setSize(450,290);
                setVisible(false);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Evolution Process");
                getContentPane().add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(5,70,110,22);

        JLabel10.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel10.setText("Step Type");
                getContentPane().add(JLabel10);
                JLabel10.setForeground(java.awt.Color.black);
                JLabel10.setBounds(5,110,110,22);

        JLabel12.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel12.setText("Step Version");
                getContentPane().add(JLabel12);
                JLabel12.setForeground(java.awt.Color.black);
                JLabel12.setBounds(5,150,110,22);
                OKButton.setText("OK");
                OKButton.setActionCommand("jbutton");
                getContentPane().add(OKButton);
                OKButton.setBounds(148,210,75,22);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("jbutton");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(226,210,75,22);
                getContentPane().add(stepIDComboBox);
                stepIDComboBox.setBounds(117,110,300,22);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel1.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel1.setText("Open Step Version: ");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel1.setBounds(0,6,180,25);

        projectLabel.setVerticalAlignment(com.sun.java.swing.SwingCon
stants.BOTTOM);
                projectLabel.setText("Project Label");
                getContentPane().add(projectLabel);
                projectLabel.setForeground(java.awt.Color.black);
                projectLabel.setFont(new Font("Dialog", Font.BOLD,
18));
                projectLabel.setBounds(187,6,200,25);
                getContentPane().add(EHLComboBox);
                EHLComboBox.setBounds(117,70,300,22);
                getContentPane().add(versionComboBox);
                versionComboBox.setBounds(117,150,300,22);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                SymItem lSymItem = new SymItem();
                EHLComboBox.addItemListener(lSymItem);
                stepIDComboBox.addItemListener(lSymItem);
                //}}
        }
```

```
  /**
   * To be called when Open Step Version Menu Item of CasesFrame
   *     receives the event from user.
   * Retrieving VersionControl and EHL object from current.cfg and
loop.cfg files
   */
  public AVCOpenStepFrame( String dirName, String pathName ){
    this();
          this.projectLabel.setText( dirName );
          this.pathName = pathName;

    try{
      FileInputStream fileInput = new FileInputStream(
this.pathName+"\\loop.cfg" ) ;
      ObjectInputStream loopIn = new ObjectInputStream( fileInput );
      if( loopIn != null ){
        Vector loopVector = new Vector();
        loopVector = (Vector)loopIn.readObject();
        if( loopVector.size() > 0 ){
          this.setLoopNameComboBox( loopVector );
        }
      }
      loopIn.close();
      fileInput.close();
    }
    catch( IOException e ){
      debug("IOException_loop: "+e);
    }
    catch( ClassNotFoundException ex ){
      debug("ClassNotFoundException_loop: "+ex);
    }

    try{
      FileInputStream fileInput = new FileInputStream(
this.pathName+"\\current.vsn" );
      ObjectInputStream currentIn = new ObjectInputStream( fileInput );
      if( currentIn != null ){
        VersionControl oldVersionControl =
(VersionControl)currentIn.readObject();
        if( oldVersionControl != null ){
          this.setInitialFrame(oldVersionControl);
        }
      }
      currentIn.close();
      fileInput.close();
    }
    catch( IOException e ){
      debug("IOException_currentIn: "+e);
    }
    catch( ClassNotFoundException ex ){
      debug("ClassNotFoundException_currentLoop: "+ex);
    }
  }

    public AVCOpenStepFrame(String sTitle)
    {
        this();
        setTitle(sTitle);
    }

    public void setVisible(boolean b)
    {
        if (b)
            setLocation(50, 50);
        super.setVisible(b);
    }

    static public void main(String args[])
    {
        (new AVCOpenStepFrame()).setVisible(true);
    }

    public void addNotify()
    {
```

```
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel10 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel12 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JComboBox stepIDComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel projectLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox EHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox versionComboBox = new
com.sun.java.swing.JComboBox();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                }
        }

  /**
   * Create new directory with new version number for all steps of the
current project
   * @param event, occur when user press OK button
   */
  public void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
```

```
        String currentLoop = (String)this.EHLComboBox.getSelectedItem();
        String currentStep = (String)this.stepIDComboBox.getSelectedItem();
        String currentVersion = (String)
this.versionComboBox.getSelectedItem();
        String currentStatus = "";

        VersionControl vc = new VersionControl( currentLoop, currentStep,
                                currentVersion, currentStatus );

        try{
          FileOutputStream fileOutput = new FileOutputStream(
this.pathName+"\\current.vsn" );
          ObjectOutputStream currentOut= new ObjectOutputStream(
fileOutput );
                if( currentOut != null ){
                  currentOut.writeObject( vc );
                }
                currentOut.flush();
                currentOut.close();
          fileOutput.close();
        }
        catch( IOException e1 ){
          debug("IOException_currentOut: "+e1);
        }

        setVisible( false );
        dispose();
            }

    /**
     * Exit AVCOpenStepFrame
     * @param event, occur when user press Cancel button
     */
          public void
cancelButton_actionPerformed(java.awt.event.ActionEvent event)
          {
        setVisible( false );
        dispose();
          }

          class SymItem implements java.awt.event.ItemListener
          {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == EHLComboBox)

          EHLComboBox_itemStateChanged(event);
                        else if (object == stepIDComboBox)

          stepIDComboBox_itemStateChanged(event);
                }
          }

    /**
     * Allows a user to select all the available Evolution processes in the
combobox
     * @param event, occur when user select Evolution Process
     */
          public void
EHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
          {
                if( event.getStateChange() == ItemEvent.SELECTED ){
          String selectedItem = (String)event.getItem();
          int selectedIndex = this.EHLComboBox.getSelectedIndex();

        if( selectedIndex > 0 ){
          if( this.EHLHashtable.containsKey( selectedItem ) ){
            EHL ehl = (EHL)this.EHLHashtable.get( selectedItem );

              StringTokenizer st = new StringTokenizer(
(String)ehl.getEHLPath(), "," );
                  Vector tokenizeVector = new Vector();
```

```
                while( st.hasMoreTokens() ){
                  tokenizeVector.addElement( st.nextToken() );
                }
              this.setStepComboBox( tokenizeVector );
            }
          }
          else if( selectedIndex == 0 ){
            this.stepIDComboBox.removeAllItems();
            this.versionComboBox.removeAllItems();
          }
        }
      }

      /**
       * Allows a user to select all the available step ID in the combobox
       * @param event, occur when user select step ID
       */
            public void
      stepIDComboBox_itemStateChanged(java.awt.event.ItemEvent event)
            {
                  if( event.getStateChange() == ItemEvent.SELECTED ){
          String selectedStep = ((String)event.getItem()).trim();
          int selectedIndex = this.stepIDComboBox.getSelectedIndex();

          if( selectedIndex > 0 ){

            Vector versionVector = new Vector();
            File existedDir = new File( this.pathName, selectedStep );
            if( existedDir.isDirectory() ){
              String[] fileList = existedDir.list();

              for( int j=0; j<fileList.length; j++ ){
                versionVector.addElement( fileList[j] );
              }
              this.setVersionComboBox(versionVector);
            }
                }
          else if( selectedIndex == 0 ){
            this.versionComboBox.removeAllItems();
          }
                }
      }

      /**
       * Short cut to print the output
       * @param string : the output string
       */
            public void debug( String string ){
              System.out.println( string );
            }

      /**
       * Adding evolution processes into EHLComboBox
       * @param loopVector : vector of evolution processes
       */
      public void setLoopNameComboBox( Vector loopVector ){
        EHLComboBox.removeAllItems();
        EHLComboBox.addItem(PROCESS_TITLE);

        this.EHLHashtable = new Hashtable();

        for( int i=0; i< loopVector.size(); i++ ){
          EHL ehl = (EHL)loopVector.elementAt( i );

          //set step Hashtable
          this.EHLHashtable.put( ehl.getEHLName(), ehl );

          this.EHLComboBox.addItem(ehl.getEHLName());
        }
            }

      /**
       * Adding step ID into stepIDComboBox
       * @param stepVector : vector of step ID
```

```
    */
    public void setStepComboBox( Vector stepVector ){
        this.stepIDComboBox.removeAllItems();
        this.stepIDComboBox.addItem(STEP_TYPE_TITLE);

        Vector versionVector = new Vector();
        for( int i=0; i< stepVector.size(); i++ ){
            String stepID = (String)stepVector.elementAt( i );
            this.stepIDComboBox.addItem( stepID.trim() );
        }

        this.setVersionComboBox( versionVector );
    }

    /**
     * Adding version numbers to versionComboBox
     * @param versionVector : vector of version numbers
     */
    public void setVersionComboBox( Vector versionVector ){
        this.versionComboBox.removeAllItems();

        for( int i=0; i < versionVector.size(); i++ ){
            String vv = (String)versionVector.elementAt( i );
            this.versionComboBox.addItem( vv.trim() );
        }
    }

    /**
     * Adding version numbers to currentStepComboBox and
mergedStepComboBox
     * @param vc : vector of VersionControl objects
     */
    public void setInitialFrame( VersionControl vc ){
        if( this.EHLComboBox.getItemCount() > 0 ){
            this.EHLComboBox.setSelectedItem(vc.getCurrentLoop());
            this.stepIDComboBox.setSelectedItem(vc.getCurrentStep());
            this.versionComboBox.setSelectedItem(vc.getCurrentVersion());
        }
    }
}
```

```
package Cases;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import com.sun.java.swing.*;
import java.text.*;

////////////////////////////////////////////////////////////////////////
/**
 * Automatic Version Control - Evolution History Splitting : to split the
existing
 *     versions into new version.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_AVC
 */
////////////////////////////////////////////////////////////////////////
public class AVCSplittingFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_AVC
{
    /**
     * pathName : current path name, e.g. C:\Cases\projectName,
D:\Cases\projectName, ...
     */
        public String pathName;

    /**
```

```
    * versionVector : stores all version numbers of current project
    */
   public Vector versionVector = new Vector();

   /**
    * EHLHashtable : stores all EHL objects which retrieve from loop.cfg
file
    */
         public Hashtable EHLHashtable = new Hashtable();

  /**
    * depHashtable : stores all Dependency objects which retrieve from
dependency.cfg file
    */
   public Hashtable depHashtable = new Hashtable();


  /**
    * Creating AVCSplittingFrame
    */
         public AVCSplittingFrame()
         {
                  // This code is automatically generated by Visual Cafe
when you add
                  // components to the visual environment. It instantiates
and initializes
                  // the components. To modify the code, only use code
syntax that matches
                  // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                  // parse your Java file into its visual environment.
                  //{{INIT_CONTROLS
                  setTitle("Automated Version Control -  Evolution History
Splitting");
                  getContentPane().setLayout(null);
                  setSize(480,280);
                  setVisible(false);
```

```
         JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                  JLabel9.setText("Evolution Process");
                  getContentPane().add(JLabel9);
                  JLabel9.setForeground(java.awt.Color.black);
                  JLabel9.setBounds(15,70,145,22);

         currentLabel.setHorizontalAlignment(com.sun.java.swing.SwingC
onstants.RIGHT);
                  currentLabel.setText("Current Step Version");
                  getContentPane().add(currentLabel);
                  currentLabel.setForeground(java.awt.Color.black);
                  currentLabel.setBounds(15,110,145,22);
                  OKButton.setText("OK");
                  OKButton.setActionCommand("jbutton");
                  getContentPane().add(OKButton);
                  OKButton.setBounds(163,210,75,22);
                  cancelButton.setText("Cancel");
                  cancelButton.setActionCommand("jbutton");
                  getContentPane().add(cancelButton);
                  cancelButton.setBounds(241,210,75,22);
                  newVersionTextField.setEditable(false);
                  getContentPane().add(newVersionTextField);

         newVersionTextField.setBackground(java.awt.Color.white);

         newVersionTextField.setForeground(java.awt.Color.black);
                  newVersionTextField.setBounds(164,150,300,22);
                  getContentPane().add(currentStepComboBox);
                  currentStepComboBox.setBounds(164,110,300,22);

         JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);

         JLabel1.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
```

```
                JLabel1.setText("Evolution History Splitting:");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel1.setBounds(0,6,250,25);

        projectLabel.setVerticalAlignment(com.sun.java.swing.SwingCon
stants.BOTTOM);
                projectLabel.setText("Project Label");
                getContentPane().add(projectLabel);
                projectLabel.setForeground(java.awt.Color.black);
                projectLabel.setFont(new Font("Dialog", Font.BOLD,
18));
                projectLabel.setBounds(250,6,230,25);
                getContentPane().add(EHLComboBox);
                EHLComboBox.setBounds(164,70,300,22);
                newStepVersionButton.setText("New Step Version");
                newStepVersionButton.setActionCommand("New Step
Version");
                getContentPane().add(newStepVersionButton);
                newStepVersionButton.setBounds(25,150,135,22);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                SymItem lSymItem = new SymItem();
                currentStepComboBox.addItemListener(lSymItem);
                EHLComboBox.addItemListener(lSymItem);
                newStepVersionButton.addActionListener(lSymAction);
                //}}
        }
```

```
    /**
     * To be called when Evolution History Splitting Menu Item of
CasesFrame
     *      receives the event from user.
     * Retrieving Dependency and EHL object from dependency.cfg and
loop.cfg files
     */
          public AVCSplittingFrame(String pathName, String dirName ){
        this();
              this.projectLabel.setText( dirName );
              this.pathName = pathName;
        try{
          FileInputStream fileInput = new FileInputStream(
this.pathName+"\\dependency.cfg" );
          ObjectInputStream dep = new ObjectInputStream( fileInput );
          if( dep != null ){
            this.depHashtable = (Hashtable) dep.readObject();
          }
          dep.close();
          fileInput.close();
        }
        catch( IOException e ){
          debug("IOException_Dep: "+e);
        }
        catch( ClassNotFoundException ex ){
          debug("ClassNotFoundException_Dep: "+ex);
        }

        try{
          FileInputStream fileInput = new FileInputStream(
this.pathName+"\\loop.cfg" );
          ObjectInputStream loopIn = new ObjectInputStream( fileInput );
          if( loopIn != null ){
            Vector loopVector = new Vector();
            loopVector = (Vector)loopIn.readObject();
            if( loopVector.size() > 0){
              this.setLoopNameComboBox( loopVector );
```

```
            }
        }
        loopIn.close();
        fileInput.close();
    }
    catch( IOException e ){
        debug("IOException_loop: "+e);
    }
    catch( ClassNotFoundException ex ){
        debug("ClassNotFoundException_loop: "+ex);
    }
        }

        public AVCSplittingFrame(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new AVCSplittingFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();
                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel currentLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField newVersionTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox currentStepComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
```

```
        com.sun.java.swing.JLabel projectLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox EHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton newStepVersionButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                        else if (object == newStepVersionButton)

        newStepVersionButton_actionPerformed(event);
                }
        }

  /**
    * Create new directory with new version number for all steps of the
current project
    * @param event, occur when user press OK button
    */
        public void
OKButton_actionPerformed(java.awt.event.ActionEvent event)
        {
     checkPath(this.versionVector);
            setVisible( false );
     dispose();
        }

  /**
    * Exit AVCSplittingFrame
    * @param event, occur when user press Cancel button
    */
        public void
cancelButton_actionPerformed(java.awt.event.ActionEvent event)
        {
     setVisible( false );
     dispose();
        }

        /**
          * Create new version number and the default version is 1.1
    * @param event, occur when user press New Version button
          */
        public void
newStepVersionButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          if( (EHLComboBox.getItemCount() > 0) &&
(EHLComboBox.getSelectedIndex() > 0) ){
            if( this.currentStepComboBox.getItemCount() == 0 ){
              newVersionTextField.setText("1.1");
            }
            else{
              createVersionNumber();
            }
     }
        }

        class SymItem implements java.awt.event.ItemListener
        {
```

```
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == currentStepComboBox)

        currentStepComboBox_itemStateChanged(event);
                        else if (object == EHLComboBox)

        EHLComboBox_itemStateChanged(event);
                }
        }

  /**
    * Allows a user to select all the available Evolution processes in the
combobox
    * @param event, occur when user select Evolution Process
    */
        public void
EHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == ItemEvent.SELECTED ){
       String selectedItem = (String)event.getItem();
       int selectedIndex = this.EHLComboBox.getSelectedIndex();

       if( selectedIndex > 0 ){
         if( this.EHLHashtable.containsKey( selectedItem ) ){
           EHL ehl = (EHL)this.EHLHashtable.get( selectedItem );
           this.versionVector = new Vector();
           this.versionVector =
tokenizeVector((String)ehl.getEHLPath());
           this.setVersionComboBox( this.versionVector);
         }
       }
       else if( selectedIndex == 0 ){
         this.currentStepComboBox.removeAllItems();
         this.newVersionTextField.setText("");
        }
      }
    }

    /**
     * Allows a user to select all the available steps in the combobox
     * @param event, occur when user select currentStepComboBox
     */
        public void
currentStepComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == ItemEvent.SELECTED ){
       newVersionTextField.setText("");
          }
    }

    /**
     * Short cut to print the output
     * @param string : the output string
     */
        public void debug( String string ){
          System.out.println( string );
        }

    /**
     * Tokenizing a string
     * @param string : tokenized string
     * @return v : vector of string without ","
     */
        public Vector tokenizeVector(String string){
      StringTokenizer st = new StringTokenizer( string, "," );
          Vector v = new Vector();
          while( st.hasMoreTokens() ){
            v.addElement( st.nextToken() );
          }
      return v;
        }
```

```
/**
 * Adding evolution processes into EHLComboBox
 * @param loopVector : vector of evolution processes
 */
      public void setLoopNameComboBox( Vector loopVector ){
  EHLComboBox.removeAllItems();
  EHLComboBox.addItem(PROCESS_TITLE);

  this.EHLHashtable = new Hashtable();

  for( int i=0; i< loopVector.size(); i++ ){
    EHL ehl = (EHL)loopVector.elementAt( i );
    String loopName = ehl.getEHLName();

    //set step Hashtable
    this.EHLHashtable.put( loopName, ehl );

    EHLComboBox.addItem(loopName);
  }
    }

/**
 * Adding version numbers to currentStepComboBox
 * @param versionVector : vector of version numbers
 */
public void setVersionComboBox( Vector versionVector ){
  this.currentStepComboBox.removeAllItems();
  File aFile = new File(this.pathName,
(String)versionVector.elementAt(0));
  if( aFile.isDirectory() ){
    String[] list = aFile.list();

    if( list.length > 0 ){
      for( int i=0; i<list.length; i++ ){
        this.currentStepComboBox.addItem(((String)list[i]).trim());
      }
    }
  }
}


/**
 * Create new version number for all steps in the current project
 * new version = highest version + 1.1
 */
public void createVersionNumber(){
  try{
    int variantMax = findMaxVariant() + 1;
        String current =
((String)currentStepComboBox.getSelectedItem()).trim();
    int index = current.indexOf(".");
    String sub1 = current.substring(0, index);
    String sub2 = current.substring(index+1);
    int i1 = Integer.parseInt(sub1)+1;
    int i2 = Integer.parseInt(sub2)+1;
    String stepVersion = variantMax+"."+i2;
    for( int i=0; i<currentStepComboBox.getItemCount(); i++ ){
      String s = (String)currentStepComboBox.getItemAt(i);
      if( s.equals(stepVersion) ){
        JOptionPane.showMessageDialog(this, stepVersion+" version
already exists in this project!",
                      "Error Message",
JOptionPane.ERROR_MESSAGE);
        return;
      }
      else if( i==currentStepComboBox.getItemCount()-1){
        newVersionTextField.setText(stepVersion);
      }
    }
  }
  catch( Exception e){}
}
```

```
/**
 * Go to each step (e.g., s-I, s-C, s-R, ...) and create new directory with
 *  the name is new version number
 * @param v : vector of step names
 */
public void checkPath(Vector v){
   String mergedVersion = this.newVersionTextField.getText();
    if( v.size() > 0 ){
      File myFile = new File(this.pathName);
      if( myFile.exists()) {
         String[] myList = myFile.list();
         for(int m=0; m<myList.length; m++ ){
            File f = new File(myFile, (String)myList[m]);
            if( f.isDirectory() ){
               if( v.contains(f.getName()) ){
                  File theFile = new File(f,mergedVersion);
                  theFile.mkdir();
                  if( !theFile.isDirectory() ){
                     theFile.mkdir();
                  }
                  if( theFile.isDirectory() ){
                     Dependency dep = (Dependency)
this.depHashtable.get( f.getName() );
                        createFiles(dep, theFile);
                  }
               }
            }
         }
      }
   }

/**
  * Create Component Content directory and link files inside it.
  * @param dep : Dependency object of theFile and get input.p and
input.s files
  * @param theFile : current version directory
  */
public void createFiles( Dependency dep, File theFile){
   if( dep !=  null ){
      try{
         //Create Component Content subdirectory in thte new step
         //and include txt.link, data.link, url.link, and caps.link
         File compContent = new File(theFile,"Component Content");
         compContent.mkdir();
         if( compContent.isDirectory() ){
            for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
               File newFile = new File(compContent,
LINK_FILE_NAMES[i]);
               FileWriter fileWriter = new FileWriter( newFile );
               BufferedWriter bw = new BufferedWriter( fileWriter );
               bw.flush();
               bw.close();
               fileWriter.close();
            }
         }

         FileOutputStream fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.p");
         DataOutputStream depPrimary = new DataOutputStream(
fileOutput);
         if( depPrimary != null ){
            depPrimary.writeBytes(dep.getPrimaryInput());
         }
         depPrimary.flush();
         depPrimary.close();
         fileOutput.close();

         fileOutput = new
FileOutputStream(theFile.getAbsolutePath()+"\\input.s");
         DataOutputStream depSecondary = new DataOutputStream(
fileOutput);
         if( depSecondary != null ){
            depSecondary.writeBytes(dep.getSecondaryInput());
```

```
            }
            depSecondary.flush();
            depSecondary.close();
            fileOutput.close();
          }
          catch( IOException e ){
            debug("dep_IOException: "+e);
          }
        }
      }
    }
    /**
     * Find the maximum variant number of the step
     * @return variantMax : the maximum variant number of this process
     */
    public int findMaxVariant(){
      int variantMax = 0;
      File aFile = new File(this.pathName,
(String)versionVector.elementAt(0));
      if( aFile.isDirectory() ){
        String[] list = aFile.list();

        if( list.length > 0 ){
          Vector v = new Vector();
          for( int i=0; i<list.length; i++ ){
            String s = list[i];
            int index = s.indexOf(".");
            String sub1 = (s.substring(0,index)).trim();
            if( !v.contains(sub1) ){
              v.addElement(sub1);
            }
          }
          if( v.size() > 0 ){
            try{
              variantMax = Integer.parseInt((String)v.elementAt(0));
              for(int j=1; j<v.size(); j++ ){
                variantMax =
Math.max(variantMax,Integer.parseInt((String)v.elementAt(j)));
              }
            }
            catch(Exception e){}
          }
        }
      }
      return variantMax;
    }
}
```

```
package Cases;

import java.awt.*;
import java.text.*;
import java.util.*;
import com.sun.java.swing.*;
import symantec.itools.awt.util.Calendar;

////////////////////////////////////////////////////////////////////////
/**
 * CalendarDialog : display the calendar and return the selected day as a
string
 */
////////////////////////////////////////////////////////////////////////
public class CalendarDialog extends com.sun.java.swing.JDialog
{
  private int index = 0;
  StepContentFrame scf = null;

    public CalendarDialog(Frame parent)
    {
        super(parent);
```

```
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Calendar");
                setModal(true);
                getContentPane().setLayout(null);
                setSize(220,250);
                setVisible(false);
                getContentPane().add(theCalendar);
                theCalendar.setBackground(java.awt.Color.lightGray);
                theCalendar.setFont(new Font("Dialog", Font.BOLD,
10));
                theCalendar.setBounds(5,15,210,170);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("Cancel");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(109,200,73,24);
                OKButton.setText("OK");
                OKButton.setActionCommand("OK");
                getContentPane().add(OKButton);
                OKButton.setBounds(37,200,73,24);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                //}}
        }
```

```
  /**
   * Receive the event from Deadline, Earliest Start Time, and Finish Time
buttons
   * from StepContentFrame
   */
  public CalendarDialog(StepContentFrame scf, String currentDate, int
index){
    this();
    this.scf = scf;
    this.index = index;

    try{
      theCalendar.setDate(currentDate);
    }
    catch(Exception e){System.out.println(e);}
  }

        public CalendarDialog()
        {
                this((Frame)null);
        }

        public CalendarDialog(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)

        setLocation((Toolkit.getDefaultToolkit().getScreenSize().width -
                                this.getSize().width) / 2,

        (Toolkit.getDefaultToolkit().getScreenSize().height -
                                this.getSize().height) / 2);
```

```
            super.setVisible(b);
        }

        static public void main(String args[])
        {
            (new CalendarDialog()).setVisible(true);
        }

        public void addNotify()
        {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                return;
            frameSizeAdjusted = true;

            // Adjust size of frame according to the insets
            Insets insets = getInsets();
            setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        symantec.itools.awt.util.Calendar theCalendar = new
symantec.itools.awt.util.Calendar();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        //}}
```



```
        class SymAction implements java.awt.event.ActionListener
        {
            public void actionPerformed(java.awt.event.ActionEvent
event)
            {
                Object object = event.getSource();
                if (object == OKButton)
                    OKButton_actionPerformed(event);
                else if (object == cancelButton)
                    cancelButton_actionPerformed(event);
            }
        }

    /**
     * Return the selected day to the specific textfield which is respectively
with
     * the selected button
     * If index = 0 : set the selected day to earliestSTTextField
     * If index = 1 : set the selected day to deadlineTextField
     * If index = 2 : set the selected day to finishTimeTextField
     * @param event : button action event
     */
        void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      if( this.scf != null ){
        if( this.index == 0 ){
          this.scf.earliestSTTextField.setText(theCalendar.getDate());
        }
        else if( this.index == 1 ){
          this.scf.deadlineTextField.setText(theCalendar.getDate());
        }
        else if( this.index == 2 ){
          this.scf.finishTimeTextField.setText(theCalendar.getDate());
        }
```

```
    }
    setVisible(false);
    dispose();
        }

  /**
   * Exit CalendarDialog and return null
   */
        void cancelButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    setVisible(false);
    dispose();
        }
}
```



```
package Cases;

/**
 * The main CASES frame.
 */

import JobSchedule.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import com.sun.java.swing.preview.JFileChooser;
import java.io.*;
import java.util.*;
```

```
////////////////////////////////////////////////////////////////////
/**
 * CasesFrame : main frame of Cases.  It is the main frame to connect to all
 * frames in Cases system.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_Cases
 */
////////////////////////////////////////////////////////////////////
public class CasesFrame extends com.sun.java.swing.JFrame implements
CasesTitle, I_Cases
{
        /**
         * fileChooser : instantiate new JFileChooser with the current
directory is Cases
         */
        public JFileChooser fileChooser = new
JFileChooser(CASESDIRECTORY);

  /**
   * projectAtomicVector : a vector contains all atomics in the project.
   * It is used in Job Schedule.
   */
  public Vector projectAtomicVector = new Vector();

  /**
   * projectStepContentVector : a vector contains all StepContent objects
in the project
   * It is used in Job Schedule
   */
  public Vector projectStepContentVector = new Vector();

  /**
   * stepContentPathVector : a vector containts the paths of all step
contents in the project
   * It is used in Job Schedule
```

```
    */
    public Vector stepContentPathVector = new Vector();

    /**
     * personnelVector : a vector contains all Personnel objects in the project
     * It is used in Job Schedule
     */
    public Vector personnelVector = new Vector();

    /**
     * scheduleAtomicVector : a vector contains all StepContent objects
whose
     * status are "Scheduled"
     * It is used in Job Schedule
     */
    public Vector scheduledAtomicVector = new Vector();

    /**
     * vc : a VersionControl object
     */
    public VersionControl vc = null;

    /**
     * title : a title of the current menu item under SPIDER menu,
     * e.g. Edit, Decompose, Component Content, Step Content, or Trace
     */
    public String title = null;

    /**
     * projectName : a current project name, e.g. C4I, C3I, ...
     */
    public String projectName = null;

    /**
     * pathName : the absolute path of the current project, e.g. C:\Cases\C4I
     */
    public String pathName = null;

    ///Job Schedule Variables
    public Vector person_queue;
    public Vector job_queue = new Vector();
    public Vector job_queue1 = new Vector();
    public Vector job_queue2 = new Vector();
    public Vector job_pool = new Vector();
    public int personid=0;
    public int ctrl=0;

    /**
     * Create CasesFrame
     */
        public CasesFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setJMenuBar(casesMenuBar);
                setTitle("CASES");
                getContentPane().setLayout(null);
                getContentPane().setForeground(java.awt.Color.blue);
                setSize(550,200);
                setVisible(false);
                //$$ casesMenuBar.move(0,348);
                projectMenu.setText("Project");
                projectMenu.setActionCommand("Hypergraph");
                projectMenu.setMnemonic((int)'P');
                casesMenuBar.add(projectMenu);
                createProjectMenuItem.setText("Create Project");
```

```
createProjectMenuItem.setActionCommand("Create
Project");
createProjectMenuItem.setMnemonic((int)'C');
projectMenu.add(createProjectMenuItem);
openProjectMenuItem.setText("Open Project");
openProjectMenuItem.setActionCommand("Open
Project");
openProjectMenuItem.setMnemonic((int)'O');
projectMenu.add(openProjectMenuItem);
deleteProjectMenuItem.setText("Delete Project");
deleteProjectMenuItem.setActionCommand("Delete
Project");
deleteProjectMenuItem.setMnemonic((int)'D');
projectMenu.add(deleteProjectMenuItem);
projectMenu.add(JSeparator1);
exitMenuItem.setText("Exit");
exitMenuItem.setActionCommand("Exit");
exitMenuItem.setMnemonic((int)'X');
projectMenu.add(exitMenuItem);
AVCMenu.setText("Automated Version Control");
AVCMenu.setActionCommand("Automated Version
Control");
AVCMenu.setMnemonic((int)'A');
casesMenuBar.add(AVCMenu);
AVCCreationMenuItem.setText("Create Step Version");
AVCCreationMenuItem.setActionCommand("New
Step");
AVCCreationMenuItem.setMnemonic((int)'C');
AVCMenu.add(AVCCreationMenuItem);
AVCOpenStepMenuItem.setText("Open Step Version");
AVCOpenStepMenuItem.setActionCommand("Open
Step");
AVCOpenStepMenuItem.setMnemonic((int)'O');
AVCMenu.add(AVCOpenStepMenuItem);
AVCMenu.add(JSeparator3);
AVCSplittingMenuItem.setText("Evolution History
Splitting");
AVCSplittingMenuItem.setActionCommand("New
Step");
AVCSplittingMenuItem.setMnemonic((int)'S');
AVCMenu.add(AVCSplittingMenuItem);
AVCMergingMenuItem.setText("Evolution History
Merging");
AVCMergingMenuItem.setActionCommand("New
Step");
AVCMergingMenuItem.setMnemonic((int)'M');
AVCMenu.add(AVCMergingMenuItem);
spiderMenu.setText("SPIDER");
spiderMenu.setActionCommand("Editor");
spiderMenu.setMnemonic((int)'S');
casesMenuBar.add(spiderMenu);
editMenuItem.setText("Edit");
editMenuItem.setActionCommand("jmenuItem");
editMenuItem.setMnemonic((int)'E');
spiderMenu.add(editMenuItem);
decomposeMenuItem.setText("Decompose");
decomposeMenuItem.setActionCommand("jmenuItem");
decomposeMenuItem.setMnemonic((int)'D');
spiderMenu.add(decomposeMenuItem);
spiderMenu.add(JSeparator4);
componentContentMenuItem.setText("Component
Content");
componentContentMenuItem.setActionCommand("jmenuItem");
componentContentMenuItem.setMnemonic((int)'C');
spiderMenu.add(componentContentMenuItem);
stepContentMenuItem.setText("Step Content");
stepContentMenuItem.setActionCommand("jmenuItem");
stepContentMenuItem.setMnemonic((int)'S');
spiderMenu.add(stepContentMenuItem);
spiderMenu.add(JSeparator2);
traceMenuItem.setText("Trace");
traceMenuItem.setActionCommand("jmenuItem");
traceMenuItem.setMnemonic((int)'T');
```

```
spiderMenu.add(traceMenuItem);
toolsMenu.setText("Tools");
toolsMenu.setActionCommand("Tools");
toolsMenu.setMnemonic((int)'T');
casesMenuBar.add(toolsMenu);
textEditorMenuItem.setText("Text Editor");
textEditorMenuItem.setActionCommand("Text Editor");
textEditorMenuItem.setMnemonic((int)'X');
toolsMenu.add(textEditorMenuItem);
MSWordMenuItem.setText("MS Word");
MSWordMenuItem.setActionCommand("Step
Database");
MSWordMenuItem.setMnemonic((int)'W');
toolsMenu.add(MSWordMenuItem);
MSExcelMenuItem.setText("MS Excel");
MSExcelMenuItem.setActionCommand("Component
Database");
MSExcelMenuItem.setMnemonic((int)'E');
toolsMenu.add(MSExcelMenuItem);
toolsMenu.add(JSeparator5);
netscapeMenuItem.setText("Netscape");
netscapeMenuItem.setActionCommand("Netscape");
netscapeMenuItem.setMnemonic((int)'N');
toolsMenu.add(netscapeMenuItem);
capsMenuItem.setText("CAPS");
capsMenuItem.setActionCommand("CAPS");
capsMenuItem.setMnemonic((int)'C');
toolsMenu.add(capsMenuItem);
toolsMenu.add(JSeparator6);
personnelDataMenu.setText("Personnel Data");
personnelDataMenu.setActionCommand("personnel
Data");
personnelDataMenu.setMnemonic((int)'P');
toolsMenu.add(personnelDataMenu);
addPersonnelMenuItem.setText("Add");
addPersonnelMenuItem.setActionCommand("Add");
addPersonnelMenuItem.setMnemonic((int)'A');
personnelDataMenu.add(addPersonnelMenuItem);
editPersonnelMenuItem.setText("Edit");
editPersonnelMenuItem.setActionCommand("Edit");
editPersonnelMenuItem.setMnemonic((int)'E');
personnelDataMenu.add(editPersonnelMenuItem);
deletePersonnelMenuItem.setText("Delete");
deletePersonnelMenuItem.setActionCommand("Delete");
deletePersonnelMenuItem.setMnemonic((int)'D');
personnelDataMenu.add(deletePersonnelMenuItem);
jobScheduleMenu.setText("Job Schedule");
jobScheduleMenu.setActionCommand("Tools");
jobScheduleMenu.setMnemonic((int)'J');
casesMenuBar.add(jobScheduleMenu);
jobManagementMenuItem.setText("Scheduling");
jobManagementMenuItem.setActionCommand("Text
Editor");
jobManagementMenuItem.setMnemonic((int)'S');
jobScheduleMenu.add(jobManagementMenuItem);
jobAssignMenuItem.setText("Assignment");
jobAssignMenuItem.setActionCommand("Component
Database");
jobAssignMenuItem.setMnemonic((int)'A');
jobScheduleMenu.add(jobAssignMenuItem);

    imageLabel.setHorizontalAlignment(com.sun.java.swing.SwingCo
nstants.CENTER);
getContentPane().add(imageLabel);
imageLabel.setBackground(java.awt.Color.blue);
imageLabel.setBounds(18,35,125,125);
//$$ JPopupMenu1.move(24,348);
JLabel2.setText("Computer-Aided Software Evolution
System");
getContentPane().add(JLabel2);
JLabel2.setBackground(java.awt.Color.blue);
JLabel2.setForeground(java.awt.Color.blue);
JLabel2.setFont(new Font("Dialog",
Font.BOLD|Font.ITALIC, 18));
```

```
        JLabel2.setBounds(155,80,380,30);
        //$$ casesIimageIcon.move(72,348);
        //$$ mainMenuBar.move(0,348);
        //$$ JMenuBar1.move(0,348);
        //}}

        //{{INIT_MENUS
        //}}

        //{{REGISTER_LISTENERS
        SymAction lSymAction = new SymAction();
        createProjectMenuItem.addActionListener(lSymAction);
        openProjectMenuItem.addActionListener(lSymAction);
        deleteProjectMenuItem.addActionListener(lSymAction);
        exitMenuItem.addActionListener(lSymAction);

AVCOpenStepMenuItem.addActionListener(lSymAction);
        textEditorMenuItem.addActionListener(lSymAction);
        MSWordMenuItem.addActionListener(lSymAction);
        MSExcelMenuItem.addActionListener(lSymAction);
        netscapeMenuItem.addActionListener(lSymAction);
        traceMenuItem.addActionListener(lSymAction);
        editMenuItem.addActionListener(lSymAction);
        decomposeMenuItem.addActionListener(lSymAction);

componentContentMenuItem.addActionListener(lSymAction);
        stepContentMenuItem.addActionListener(lSymAction);

jobManagementMenuItem.addActionListener(lSymAction);
        jobAssignMenuItem.addActionListener(lSymAction);
        addPersonnelMenuItem.addActionListener(lSymAction);
        editPersonnelMenuItem.addActionListener(lSymAction);

deletePersonnelMenuItem.addActionListener(lSymAction);
        capsMenuItem.addActionListener(lSymAction);
        AVCCreationMenuItem.addActionListener(lSymAction);
        AVCMergingMenuItem.addActionListener(lSymAction);
        AVCSplittingMenuItem.addActionListener(lSymAction);
        //}}

        /**
         * Check and Create Cases and stakeholder directory
         */
        if( CASESDIRECTORY.isDirectory() ){
          String[] fileList = CASESDIRECTORY.list();
          if( fileList.length > 0 ){
            setOpenDelete( true );
          }
          else if( fileList.length == 0 ){
            setOpenDelete( false );
          }
        }
        else{
          CASESDIRECTORY.mkdir();
          setOpenDelete( false );
        }

        if( !STAKEHOLDER.isDirectory() ){
          STAKEHOLDER.mkdir();
        }
        menuSetEnabled(false);

        drawIcon();
    }

    /**
     * Adding cases.gif logo to CasesFrame
     */
    public void drawIcon(){
      try{
        FileInputStream imageFile = new FileInputStream("cases.gif");
        byte[] data = new byte[3000];
        imageFile.read(data);
        ImageIcon icon = new ImageIcon(data);
```

```
            imageLabel.setIcon(icon);
        }catch (Exception e){ System.out.println(e);}
    }

        public CasesFrame(String sTitle)
        {
                this();
        setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new CasesFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel imageLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JMenuBar casesMenuBar = new
com.sun.java.swing.JMenuBar();
        com.sun.java.swing.JMenu projectMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem createProjectMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem openProjectMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem deleteProjectMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator1 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenuItem exitMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenu AVCMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem AVCCreationMenuItem = new
com.sun.java.swing.JMenuItem();
```

```
        com.sun.java.swing.JMenuItem AVCOpenStepMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator3 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenuItem AVCSplittingMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem AVCMergingMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenu spiderMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem editMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem decomposeMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator4 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenuItem componentContentMenuItem =
new com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem stepContentMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator2 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenuItem traceMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenu toolsMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem textEditorMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem MSWordMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem MSExcelMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator5 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenuItem netscapeMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem capsMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JSeparator JSeparator6 = new
com.sun.java.swing.JSeparator();
        com.sun.java.swing.JMenu personnelDataMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem addPersonnelMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem editPersonnelMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem deletePersonnelMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenu jobScheduleMenu = new
com.sun.java.swing.JMenu();
        com.sun.java.swing.JMenuItem jobManagementMenuItem = new
com.sun.java.swing.JMenuItem();
        com.sun.java.swing.JMenuItem jobAssignMenuItem = new
com.sun.java.swing.JMenuItem();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == createProjectMenuItem)

        createProjectMenuItem_actionPerformed(event);
                        else if (object == openProjectMenuItem)

        openProjectMenuItem_actionPerformed(event);
                        else if (object == deleteProjectMenuItem)
```

```
deleteProjectMenuItem_actionPerformed(event);
                else if (object == exitMenuItem)
                        exitMenuItem_actionPerformed(event);
                else if (object == AVCOpenStepMenuItem)

AVCOpenStepMenuItem_actionPerformed(event);

                else if (object == AVCCreationMenuItem)

AVCCreationMenuItem_actionPerformed(event);
                else if (object == AVCMergingMenuItem)

AVCMergingMenuItem_actionPerformed(event);
                else if (object == AVCSplittingMenuItem)

AVCSplittingMenuItem_actionPerformed(event);
                else if (object == editMenuItem)
                        editMenuItem_actionPerformed(event);
                else if (object == decomposeMenuItem)

decomposeMenuItem_actionPerformed(event);
                else if (object == componentContentMenuItem)

componentContentMenuItem_actionPerformed(event);
                else if (object == stepContentMenuItem)

stepContentMenuItem_actionPerformed(event);
                else if (object == traceMenuItem)

traceMenuItem_actionPerformed(event);
                else if (object == textEditorMenuItem)

textEditorMenuItem_actionPerformed(event);
                else if (object == MSWordMenuItem)

MSWordMenuItem_actionPerformed(event);
                else if (object == MSExcelMenuItem)

MSExcelMenuItem_actionPerformed(event);
                if (object == netscapeMenuItem)

netscapeMenuItem_actionPerformed(event);
                else if (object == capsMenuItem)

capsMenuItem_actionPerformed(event);
                else if (object == addPersonnelMenuItem)

addPersonnelMenuItem_actionPerformed(event);
                else if (object == editPersonnelMenuItem)

editPersonnelMenuItem_actionPerformed(event);
                else if (object == deletePersonnelMenuItem)

deletePersonnelMenuItem_actionPerformed(event);
                else if (object == jobManagementMenuItem)

jobManagementMenuItem_actionPerformed(event);
                else if (object == jobAssignMenuItem)

jobAssignMenuItem_actionPerformed(event);
        }
    }

/**
 * Ask a user to enter the new project name.
 * Do not allow a duplicate project name.
 * If the name is not duplicate, create the new project and connect to
 * ProjectSchemaFrame directly.
 *
 * @param event : action event from Create Project menu item
 */
```

```
        void
createProjectMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          this.projectName = JOptionPane.showInputDialog( this, "Project
Name",
                    "Create
Project",JOptionPane.INFORMATION_MESSAGE);
      if( this.projectName != null ){
        String[] theList = CASESDIRECTORY.list();
        if( theList.length == 0 ){
          this.setOpenDelete( true );
                File newFile = new
File(CASESDIRECTORY,this.projectName);
                newFile.mkdir();
          if( newFile.isDirectory() ){
                        this.pathName = newFile.getAbsolutePath();
            (new ProjectSchemaFrame(this.projectName,
this.pathName)).setVisible(true);
            menuSetEnabled(true);
          }
        }
        else if( theList.length>0 ){
          for( int i=0; i<theList.length; i++ ){

            // Checking the duplication before create the new project
            if( this.projectName.equals(theList[i]) ){
              i= theList.length;
              JOptionPane.showMessageDialog(this, this.projectName+"
has been created.",
                          "Error Message",
JOptionPane.ERROR_MESSAGE);
              return;
            }
            else if( i==(theList.length-1) ){
              this.setOpenDelete( true );
                File newFile = new
File(CASESDIRECTORY,this.projectName);
                newFile.mkdir();
              if( newFile.isDirectory() ){
                    this.pathName = newFile.getAbsolutePath();
                (new ProjectSchemaFrame(this.projectName,
this.pathName)).setVisible(true);
                menuSetEnabled(true);
              }
            }
          }
        }
      }

  /**
    * Allow a user to select a project, then he has a choice
    * Yes : connect to ProjectSchemaFrame to edit step.cfg, component.cfg,
loop.cfg,
    * or dependency.cfg file.
    * No : go straight to main menu (Automatic Version Control, SPIDER,
Tools, or Job Schedule
    *
    * @param event : action event from Open Project menu item
    */
        void
openProjectMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
            int returnValue=-1;
          if( this.projectName != null ){
            File newFile = new
File(CASESDIRECTORY,this.projectName);
            newFile.mkdir();
            this.fileChooser = new JFileChooser(newFile);
          }
          this.fileChooser.setDialogTitle("Open Project");
      this.fileChooser.setCurrentDirectory(CASESDIRECTORY);
```

```
this.fileChooser.setFileSelectionMode(this.fileChooser.DIRECTORIES_O
NLY);
            repaint();
      returnValue = this.fileChooser.showDialog(this,"Open");
            if( returnValue == fileChooser.APPROVE_OPTION ){

                  File file = fileChooser.getCurrentDirectory();

                  this.pathName = file.getAbsolutePath();
        int i = JOptionPane.showConfirmDialog(this, "Do you want to open
Project Schema?",
              "Confirmation", JOptionPane.YES_NO_OPTION);
        if( i==0 ){
                  (new ProjectSchemaFrame(this.projectName,
this.pathName)).setVisible(true);
                  }
        menuSetEnabled(true);
      }
         }

  /**
   * Connect to DeleteDialog, where allows a user to delete a project
   * and all existing project names are in the combo box.
   * Except, the current project can not be deleted since it is occuping.
   *
   * @param event : action event from Delete Project menu item
   */
        void
deleteProjectMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
     (new DeleteDialog(this, "Delete Project")).setVisible(true);
        }

  /**
   * Exit CasesFrame without saving or any notice
   *
   * @param event : action event from Exit menu item
   */
        void exitMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
              System.exit(0);
              dispose();
        }

  /**
   * Connect to AVCCreateStepFrame
   *
   * @param event : action event from Create Step Version
   */
        void
AVCCreationMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
              (new AVCCreateStepFrame(this.pathName,
this.projectName)).setVisible(true);
        }

  /**
   * Connect to AVCOpenStepFrame
   *
   * @param event : action event from Open Step Version
   */
        void
AVCOpenStepMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
              (new AVCOpenStepFrame(this.projectName,
this.pathName)).setVisible(true);
        }

  /**
```

```
    * Connect to AVCEHSplittingFrame
    *
    * @param event : action event from Evolution History Splitting
    */
        void
AVCSplittingMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
                (new AVCSplittingFrame(this.pathName,
this.projectName)).setVisible(true);
        }

   /**
    * Connect to AVCEHMergingFrame
    *
    * @param event : action event from Evolution History Merging
    */
        void
AVCMergingMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
                (new AVCMergingFrame(this.pathName,
this.projectName)).setVisible(true);
        }

   /**
    * Get the current versionControl object and will connect to
EditDecomposeFrame
    *
    * @param event : action event from Edit menu item
    */
        void editMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          this.title = EDIT_TITLE;
          getVersionControl();
        }

   /**
    * Get the current versionControl object and will connect to
EditDecomposeFrame
    *
    * @param event : action event from Decompose menu item
    */
        void
decomposeMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
          this.title = DECOMPOSE_TITLE;
          getVersionControl();
        }

   /**
    * Get the current versionControl object and will connect to
ComponentContentFrame
    *
    * @param event : action event from Component Content menu item
    */
        void
componentContentMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          this.title = COMPONENT_CONTENT_TITLE;
          getVersionControl();
        }

   /**
    * Get the current versionControl object and will connect to
StepContentFrame
    *
    * @param event : action event from Step Content menu item
    */
        void
stepContentMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
```

```
            this.title = STEP_CONTENT_TITLE;
            getVersionControl();
        }

    /**
     * Get the current versionControl object and will connect to TraceFrame
     *
     * @param event : action event from Trace menu item
     */
        void traceMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
            this.title = TRACE_TITLE;
            getVersionControl();
        }

    /**
     * Connect to notepad editor
     *
     * @param event : action event from Text menu item
     */
        void
textEditorMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
      try {
        Runtime runtime = Runtime.getRuntime();
        runtime.exec(NOTEPAD);
      }
      catch( Exception e ) {
        throw new RuntimeException(e.toString());
      }
        }

    /**
     * Connect to Microsoft Word editor
     *
     * @param event : action event from MS Word menu item
     */
        void
MSWordMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
      try {
        Runtime runtime = Runtime.getRuntime();
        runtime.exec(WINWORD);
      }
      catch( Exception e ) {
        throw new RuntimeException(e.toString());
      }
        }

    /**
     * Connect to Microsoft Excel editor
     *
     * @param event : action event from MS Excel menu item
     */
        void
MSExcelMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
      try {
        Runtime runtime = Runtime.getRuntime();
        runtime.exec(EXCEL);
      }
      catch( Exception e ) {
        throw new RuntimeException(e.toString());
      }
        }

    /**
     * Connect to Netscape environment
     *
     * @param event : action event from Netscape menu item
     */
        void
netscapeMenuItem_actionPerformed(java.awt.event.ActionEvent event)
```

```
        {
    try {
       Runtime runtime = Runtime.getRuntime();
       runtime.exec(NETSCAPE);
    }
    catch( Exception e ) {
       throw new RuntimeException(e.toString());
    }
        }

  /**
    * Connect to Caps environment
    *
    * @param event : action event from Caps menu item
    */
        void capsMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
    try {
       Runtime runtime = Runtime.getRuntime();
       runtime.exec(CAPS);
    }
    catch( Exception e ) {
       throw new RuntimeException(e.toString());
    }
        }

  /**
    * Connect to PersonnelFrame
    *
    * @param event : action event from Personnel - Add menu item
    */
        void
addPersonnelMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          (new PersonnelFrame()).setVisible(true);
        }

  /**
    * Go to JFileChooser to choose a file before connect to PersonnelFrame
    *
    * @param event : action event from Personnel - Edit menu item
    */
        void
editPersonnelMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
    FileDialog fd = new FileDialog(this, "Open", FileDialog.LOAD);
    fd.setDirectory(STAKEHOLDER.getAbsolutePath());
    fd.show();
    String fileName = fd.getFile();
    if( fileName != null ){
      String filePath = STAKEHOLDER.getAbsolutePath()+fileName;
        (new PersonnelFrame(filePath, "Edit")).setVisible(true);
    }
        }

  /**
    * Connect to DeleteDialog and it has a Browse button to select a file
    * under stakeholder directory
    *
    * @param event : action event from Personnel - Delete menu item
    */
        void
deletePersonnelMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          (new DeleteDialog(this, "Delete Personnel
Data")).setVisible(true);
        }

  /**
    * Short cut to print the output
```

```
     * @param string : the output string
     */
          public void debug( String string ){
             System.out.println(string);
          }

    /**
     * Gray out Open Project and Delete Project menu items
     *    if there is no project under CASESDIRECTORY .
     * Or not gray out if CASESDIRECTORY exists at least one project
     */
          public void setOpenDelete( boolean enabled ){
             this.openProjectMenuItem.setEnabled( enabled );
             this.deleteProjectMenuItem.setEnabled( enabled );
          }

    /**
     * If this.projectName is null, flag = false
     * Else flag = true
     */
          void menuSetEnabled( boolean flag){
       this.AVCMenu.setEnabled( flag );
             this.spiderMenu.setEnabled( flag );
             this.toolsMenu.setEnabled( flag );
             this.jobScheduleMenu.setEnabled(flag);
          }

    /**
     * Get VersionControl object from current.vsn file
     */
          public void getVersionControl(){
       try{
          FileInputStream fileInput = new
FileInputStream(this.pathName+"\\current.vsn");
                ObjectInputStream current = new ObjectInputStream(
fileInput );
                if( current != null ){
                   this.vc = (VersionControl) current.readObject();
                }
                current.close();
                fileInput.close();
             }
                catch( ClassNotFoundException e ){
debug("ClassNotFoundException: "+e); }
                catch(IOException i){
          JOptionPane.showMessageDialog(this, "Can not open the
application \nsince current.vsn does not exist in this project.",
                              "Error Message",
JOptionPane.ERROR_MESSAGE);
          return;
                }

                if( this.vc != null ){
          directoryTree();
       }
          }

     /**
      * Check and locate the current step in fileChooser
      * If the selected step is not a current step, Error Message shows up
      */
           public void directoryTree(){
              String currentStep = (String)this.vc.getCurrentStep();
              String currentVersion = (String)this.vc.getCurrentVersion();
              String wholePath = this.pathName +
"\\"+currentStep+"\\"+currentVersion;
              File file = new File(wholePath);
                    int returnValue=-1;
        this.fileChooser = new JFileChooser(file);
              this.fileChooser.setDialogTitle("Directory Tree");
        this.fileChooser.setCurrentDirectory(file);

this.fileChooser.setFileSelectionMode(this.fileChooser.DIRECTORIES_O
NLY);
```

```
        returnValue = this.fileChooser.showDialog(this,"Open");

            if( returnValue == this.fileChooser.APPROVE_OPTION ){

                File aFile = this.fileChooser.getCurrentDirectory();


                String absPath = aFile.getAbsolutePath();
                String output =
currentStep.substring((int)currentStep.indexOf("-")+1)+currentVersion;
                if( absPath.equals( wholePath ) ){
                  currentDir(absPath);
                }
                else if( absPath.length() > wholePath.length() ){
                  subDir(absPath);
                }
                else{
            JOptionPane.showMessageDialog(this, "The selected directory is
not the current step.",
                    "Error Message",
JOptionPane.ERROR_MESSAGE);
                }
        }
          }

    /**
     * Locate the selected step and it is an original step
     *
     * @param absPath : a string of the whole path
     */
    public void currentDir(String absPath ){
            String currentStep = (String)this.vc.getCurrentStep();
            String currentVersion = (String)this.vc.getCurrentVersion();
                String stepName = currentStep + currentVersion;
            String output =
currentStep.substring((int)currentStep.indexOf("-")+1)+currentVersion;
                if( this.title.equals(EDIT_TITLE) ){
                  (new EditDecomposeFrame(this.title,stepName,
output,absPath)).setVisible(true);
                } .
                else if( this.title.equals(DECOMPOSE_TITLE) ){
                  int theMax = (int)findMax(absPath) +1 ;
          stepName = stepName + "-" + theMax;
                  String decomposeOutput = stepName.substring(2);
                  (new EditDecomposeFrame(this.title,stepName,

decomposeOutput,absPath,theMax)).setVisible(true);
                }
                else if(
this.title.equals(COMPONENT_CONTENT_TITLE) ){
                  setComponentContent( stepName, absPath);
                }
                else if( this.title.equals(STEP_CONTENT_TITLE) ){
          setStepContent(null, stepName, absPath);
              }
                else if( this.title.equals(TRACE_TITLE) ){
          setTraceFrame(stepName, absPath);
              }
    }

    /**
     * Locate the selected step and it is a decompose step
     *
     * @param absPath : a string of the whole path
     */
    public void subDir(String absPath ){
            String currentStep = (String)this.vc.getCurrentStep();
            String currentVersion = (String)this.vc.getCurrentVersion();
            String wholePath = this.pathName +
"\\"+currentStep+"\\"+currentVersion;
              String newSubString = absPath.substring(0,
wholePath.length());
```

```
            String output =
currentStep.substring((int)currentStep.indexOf("-")+1)+currentVersion;

                    int result = wholePath.compareTo(newSubString);
                    if( result == 0 ){
                      String subName =
absPath.substring(wholePath.length()+1);
                      StringTokenizer st = new
StringTokenizer(subName,"\\");
                      Vector stVector = new Vector();
                      while( st.hasMoreTokens()){
                        stVector.addElement(st.nextToken());
                      }
                      String stResult = (String)stVector.elementAt(0);
                      for( int i=1; i<stVector.size(); i++){
                        stResult =
stResult+"."+(String)stVector.elementAt(i);
                      }
                      output = output + "-" + stResult;
                      String stepName = currentStep+currentVersion+"-
"+stResult;

                      if( this.title.equals(EDIT_TITLE) ){
                      (new EditDecomposeFrame(this.title, stepName,
output,absPath)).setVisible(true);
                      }

                      else if( this.title.equals(DECOMPOSE_TITLE) ){
            int theMax = (int)findMax(absPath) + 1;
                        stepName = stepName + "." +theMax;
                        String decomposeOutput = stepName.substring(2);
                      (new EditDecomposeFrame(this.title, stepName,
decomposeOutput, absPath, theMax)).setVisible(true);
                      }

                      else if(
this.title.equals(COMPONENT_CONTENT_TITLE) ){
                        setComponentContent( stepName, absPath);
                      }
                      else if( this.title.equals(STEP_CONTENT_TITLE) ){

                        setStepContent(null, stepName, absPath);
                      }
                      else if( this.title.equals(TRACE_TITLE) ){
              setTraceFrame(stepName, absPath);
            }
                    }
                    else{
          JOptionPane.showMessageDialog(this, "The selected directory is
not the current step.",
                        "Error Message",
JOptionPane.ERROR_MESSAGE);
                    }
    }

    /**
     * Return a vector of all components of the current step
     *
     * @param stepName : the selected step
     * @param absPath : the complete path of this stepName
     * @return componentVector : a vector of strings holds component
names
     */
    public Vector getComponents(String stepName, String absPath){
                    Vector componentVector = new Vector();
      String s = stepName.substring(2);

                    componentVector.addElement(s);
                    try{
                      FileInputStream fileInput = new
FileInputStream(absPath+"\\input.p");
                      DataInputStream inputP = new
DataInputStream(fileInput);
                      if( inputP != null ){
```

```
                    String sP = inputP.readLine();
                    if(( sP !=  null) && (!sP.equals("")) ){
                      tokenizer(componentVector, sP);
                    }
                  }
                  inputP.close();
                  fileInput.close();


                  fileInput = new FileInputStream(absPath+"\\input.s");
                  DataInputStream inputS = new
DataInputStream(fileInput);
                  if( inputS != null ){
                    String sS = inputS.readLine();
                    if( (sS != null) && (!sS.equals("")) ){
                      tokenizer(componentVector, sS);
                    }
                  }
                }
                catch( IOException e ){
                  debug("IOException at ComponentContent: "+e);
                }
    return componentVector;
  }

  /**
    * Connect to ComponentContentFrame
    *
    * @param stepName : selected step name, e.g., s-I, 1.1, 1, ...
    * @param absPath : the absolute path of  stepName, e.g.,F:\Cases\s-
I\1.1
    */
        public void setComponentContent( String stepName, String
absPath ){
        (new
ComponentContentFrame(this.pathName,stepName.substring(2),
getComponents(stepName, absPath),
(Vector)fileNameList())).setVisible(true);
        }

  /**
    * Connect to StepContentFrame
    *
    * @param traceFrame : instantiation of TraceFrame
    * @param stepName : selected step name, e.g., s-I, 1.1, 1, ...
    * @param absPath : the absolute path of  stepName, e.g.,F:\Cases\s-
I\1.1
    */
  public void setStepContent(TraceFrame traceFrame, String stepName,
String absPath ){
    File file = new File(absPath);
    if( file.isDirectory() ){
      if( !(file.getName()).equals(COMPONENT_CONTENT_DIR) ){
        Vector atomicsVector = new Vector();
        File temp = new File(this.pathName, this.vc.getCurrentStep());
        String[] theFiles = temp.list();
        for( int i=0; i<theFiles.length; i++ ){
          getAtomics(new File(temp, theFiles[i]), atomicsVector);
        }
          (new StepContentFrame(traceFrame, stepName, absPath,
                                atomicsVector )).setVisible(true);

        }
      else{
        JOptionPane.showMessageDialog(this, absPath+" is not a step.
Please reselect it!",
                      "Warning
Message",JOptionPane.ERROR_MESSAGE);
        return;
      }
    }
    else{
```

```
        JOptionPane.showMessageDialog(this, absPath+" is not a step.
Please reselect it!",
                      "Warning
Message",JOptionPane.ERROR_MESSAGE);
        return;
      }
    }

    /**
     * Connect to TraceFrame
     *
     * @param stepName : selected step name, e.g., s-I, 1.1, 1, ...
     * @param absPath : the absolute path of  stepName, e.g.,F:\Cases\s-
I\1.1
     */
    public void setTraceFrame(String stepName, String absPath){
      (new TraceFrame(this, this.pathName, getComponents(stepName,
absPath), (Vector)fileNameList())).setVisible(true);
    }

    /**
     * Get all atomics of the selected step
     *
     * @param aFile : since a file is a step, aFile is a selected step
     * @param storedVector : a vector of atomics
     */
    public void getAtomics(File aFile, Vector storedVector){
      boolean isAtomic = true;
      if( (aFile.isDirectory()) &&
!((aFile.getName()).equals(COMPONENT_CONTENT_DIR)) ){
        String[] theFiles = aFile.list();
        for( int i=0; i<theFiles.length; i++ ){
          char c = (char)((String)theFiles[i]).charAt(0);
          Character theChar = new Character(c);
          if(theChar.isDigit(c)){
            File f1 = new File( aFile, theFiles[i]);
            if( (f1.isDirectory()) &&
!((f1.getName()).equals(COMPONENT_CONTENT_DIR)) ){
              isAtomic = false;
              getAtomics(new File(aFile, theFiles[i]), storedVector );
            }
          }
        }
        if( isAtomic ){
          storedVector.addElement(aFile);
        }
      }
    }

    /**
     * Find a maximum number of the selected step version number
     *
     * @return theMax : an integer, the maximum number of the selected
step
     * @param thePath : the path of selected step, e.g., F:\Cases\s-I\1.1\1\2\1
     */
        public int findMax( String thePath ){
          Vector v = new Vector();
            File f1 = new File( thePath );
            String[] list = (String[]) f1.list();
            for(int i=0; i<list.length;i++){
              File f2 = new File(thePath, list[i]);
              if( f2.isDirectory() ){
                try{
          Integer i1 = new Integer(list[i]);
          v.addElement(i1);
                }
                catch(NumberFormatException n){
                }
        }
            }
            int theMax = 0;
            for(int i=0; i<v.size(); i++){
```

```
        theMax =
Math.max(theMax,(int)((Integer)v.elementAt(i)).intValue());
                    }
                return theMax;
            }

    /**
     * Token a string with delimit is ","
     *
     * @param v : a vector of tokenizer string
     * @param s : a tokenized string
     */
            public void tokenizer( Vector v, String s){
               StringTokenizer st = new StringTokenizer(s,",");
               while(st.hasMoreTokens()){
                  String theString = (st.nextToken()).trim();
                  v.addElement(theString);
               }
            }

    /**
     * List all steps of a project.
     * @return a vector of string with all steps
     */
            public Vector fileNameList(){
               Vector v = new Vector();
      String[] theList = (String[])(new File(this.pathName)).list();
      for( int i=0; i<theList.length;i++){
         if( (new File(this.pathName+"\\"+theList[i])).isDirectory()){
            v.addElement(theList[i]);
         }
      }
      return v;
    }

    /**
     * Save all personnel object after updating
     */
            public void savePersonnelVector(Vector v){
               personnelVector = v;
               if( personnelVector != null ){
            for( int i=0; i<personnelVector.size(); i++ ){
              Personnel personnel = (Personnel)personnelVector.elementAt(i);
              if( personnel != null ){
                 try{
                    FileOutputStream fileOutput = new
FileOutputStream(STAKEHOLDER+"\\"+personnel.getID());
                    ObjectOutputStream oo = new
ObjectOutputStream(fileOutput);
                    if( oo != null ){
                       oo.writeObject(personnel);
                    }
                    oo.flush();
                    oo.close();
                    fileOutput.close();
                 }
                 catch(IOException io){
                    JOptionPane.showMessageDialog(this, "Sorry, saving is
unsuccessful", "Error Message", JOptionPane.ERROR_MESSAGE);
                 }
              }
           }
        }
            }

    /**
     * Return a vector of personnel objects under stakeholder directory
     */
            public Vector getPersonnelVector(){
               personnelVector = new Vector();
               String[] list = (String[])STAKEHOLDER.list();
               if( list.length > 0 ){
                  for( int i=0; i<list.length; i++ ){
                        try{
```

```
            String filePath = STAKEHOLDER.getAbsolutePath()+list[i];
                        File aFile = new File(filePath);
                        if( aFile.exists() ){
                          FileInputStream fileInput =  new
FileInputStream(aFile);
                          ObjectInputStream oi = new
ObjectInputStream(fileInput);
                          if( oi != null ){
                            Personnel personnel =
(Personnel)oi.readObject();
                          if( personnel != null ){
                    personnelVector.addElement(personnel);
                            }
                          }
                          oi.close();
                          fileInput.close();
            }
                      }
                      catch( IOException io ){
                        System.out.println(io);
                      }
                      catch( ClassNotFoundException c ){
                        debug("ClassNotFoundException: "+c);
                      }
              }
            }
            return personnelVector;
          }

  /**
   * Return a vector of all atomics in the selected project
   */
        public Vector getAllAtomics(){
    File aProject = new File(this.pathName);  //current project
    projectAtomicVector = new Vector();
    if( aProject.isDirectory() ){
```

```
      String[] stepList = aProject.list();  //List all the steps of the current
project
      for( int i=0; i<stepList.length; i++ ){
        File aFile = new File(aProject,stepList[i]);  //searching for steps
only
        if( aFile.isDirectory() ){
          File temp = new File(aProject, stepList[i]);
          String[] theFiles = temp.list();
          for( int j=0; j<theFiles.length; j++ ){
            getAtomics(new File(temp, theFiles[j]),
projectAtomicVector);
          }
        }
      }
    }
        return projectAtomicVector;
      }

  /**
   * Save all stepContent objects after updating
   */
      public void saveStepContentVector(Vector v){
        projectStepContentVector = v;
        if( projectStepContentVector != null ){
          for( int i=0; i<projectStepContentVector.size(); i++ ){
       try{
          String path = (String)stepContentPathVector.elementAt(i);
          FileOutputStream fileOutput = new FileOutputStream(path);
          ObjectOutputStream oo = new
ObjectOutputStream(fileOutput);
          if( oo != null ){

oo.writeObject((StepContent)projectStepContentVector.elementAt(i));
          }
          oo.flush();
          oo.close();
          fileOutput.close();
```

```
            }
            catch(IOException io){
               debug("IOException: "+io);
            }
                }
              }
            }

/**
 * Return a vector of stepContent objects in the selected project
 */
        public Vector getStepContentVector(){
  int stepContentIndex = 0;

    stepContentPathVector = new Vector();
            projectAtomicVector = (Vector)getAllAtomics();
            projectStepContentVector = new Vector();

            if( projectAtomicVector != null ){
       for( int j=0; j<projectAtomicVector.size(); j++ ){
          File theFile = new
File((File)projectAtomicVector.elementAt(j),"step.cnt");  //Example to get
the atomic at element 4 of projectAtomicVector
          if( theFile.exists() ){
             try{
                FileInputStream fileInput = new FileInputStream(theFile);
                ObjectInputStream oo = new ObjectInputStream(fileInput);
                if( oo != null ){
                   StepContent st = (StepContent)oo.readObject();
                   if( st!= null ){
                      projectStepContentVector.insertElementAt(st,
stepContentIndex); //Step content vector of all atomics in the project

stepContentPathVector.insertElementAt(theFile.getAbsolutePath(),
stepContentIndex++);
                   }
                }
                oo.close();
                fileInput.close();
             }
             catch(IOException io){
                debug("IOException: "+io);
             }
             catch( ClassNotFoundException c ){
                debug("ClassNotFoundException: "+c);
             }
          }
       }
    }

          return projectStepContentVector;
        }

/**
 * Return a vector of atomics with the status is "Scheduled"
 */
        public Vector getScheduledAtomicVector(){
           scheduledAtomicVector = new Vector();
    if( projectStepContentVector != null ){
       for( int i=0; i<projectStepContentVector.size(); i++ ){
          StepContent st =
(StepContent)projectStepContentVector.elementAt(i);
          if( st != null ){
             String status = ((String)st.getStatus()).trim();
             if( status.equals("Approved") ){
                st.setStatus("Scheduled");
                scheduledAtomicVector.addElement(st);
             }
             else if( status.equals("Scheduled") ){
                scheduledAtomicVector.addElement(st);
             }
          }
       }
    }
```

```
    return scheduledAtomicVector;
        }
  ////////////////////////////////// JOB SCHEDULE //////////////////////////////
  /**
   * Job Management
   */
        void
jobManagementMenuItem_actionPerformed(java.awt.event.ActionEvent
event)
        {
          person_queue = (Vector)getPersonnelVector();
          job_pool = (Vector)getStepContentVector();
          for(int i=0; i<job_pool.size(); i++){
           StepContent step=(StepContent)job_pool.elementAt(i);
          }
    Predecessor();

    //cleanperson_job();//each time will kill all of person's job
    checking_person();

    if(job_queue.size()>0){
       (new JFrame_manage(job_queue)).setVisible(true);
    }
    else{
       (new JDialog_message()).setVisible(true);
    }

        }

  /**
   * Job Assignment
   */
        void
jobAssignMenuItem_actionPerformed(java.awt.event.ActionEvent event)
        {
          if(person_queue.size()>0 && job_queue.size()>0){
            (new JFrame_assignjob(this, person_queue, job_queue,
job_pool)).setVisible(true);
          }
          else{
            (new JDialog_message1()).setVisible(true);
          }
  }

        void Predecessor(){
          job_queue.removeAllElements();
          job_queue1.removeAllElements();
          job_queue2.removeAllElements();
          for(int i=0; i<job_pool.size(); i++){
            StepContent step=(StepContent)job_pool.elementAt(i);
            if(step.getStatus().equals("Approved")
||step.getStatus().equals("Scheduled")){
              job_queue1.addElement(step);
            }
          }
          for(int i=0; i<job_queue1.size(); i++){
            StepContent step=(StepContent) job_queue1.elementAt(i);
            step.setStatus("Scheduled");
            setstep(step.getStepName(), step);

            Vector v=(Vector) step.getPredecessors();

      if(v!=null){
        int n=0;
        n=checkpredecesor(step);
        if(n>0){
          this.job_queue.addElement(step);
        }
        else{
          this.job_queue2.addElement(step);
        }
      }
      else{
```

```
            this.job_queue.addElement(step);
          }
        }//end for_loop
          }

          int checkpredecesor(StepContent step){
            Vector v=(Vector) step.getPredecessors();
            if(v.size()>0){
              for(int i=0; i<v.size(); i++){
                String predecesor=(String) v.elementAt(i);
                for(int j=0; j<this.job_queue1.size(); j++){
                  StepContent step2=(StepContent)
this.job_queue1.elementAt(j);
                  if(predecesor.equals(step2.getStepName())){
                  return 0;
                  }
                }

              }//end for_loop
            }//end if
        return 1;
          }

          void checking_person(){
            for(int i=0; i<person_queue.size(); i++){
              Personnel person=(Personnel)person_queue.elementAt(i);
              if(person!=null){
                Vector major=(Vector) person.getMajorJobs();

                if(major!=null){
                  for(int j=0; j<major.size(); j++){
                    if(major!=null){
                    StepContent step=(StepContent) major.elementAt(j);
                    if(step!=null){
                      if( check_status(step.getStepName())==1){
                        major.removeElement(step);
                      }
                    }//end if(major!=null)
                  }//end for_loop(j)
                }

                //check minorJob second
                Vector minor=(Vector) person.getMinorJobs();
                if(minor!=null){
                  for(int k=0; k<minor.size(); k++){
                    StepContent step=(StepContent) minor.elementAt(k);
                    if(step!=null){
                      if( check_status(step.getStepName())==1){
                        minor.removeElement(step);
                      }
                    }
                  }//end for_loop(k)
                }
              }//end if(person!=null)

            }// end for_loop(i)
          }//end check_person

          int check_status(String stepname){
            for(int i=0; i<job_pool.size(); i++){
              if(job_pool!=null){
                StepContent step=(StepContent) job_pool.elementAt(i);
                if(step!=null){
                  if(step.getStepName().equals(stepname)){
                    if(step.getStatus().equals("Completed")){
                   return 1;
                    }
                  }
                }
              }
            }//end for_loop
            return 0;
          }
```

```
    void cleanperson_job(){
      try{
        for(int i=0; i<person_queue.size(); i++){
          Personnel p=(Personnel) person_queue.elementAt(i);
          Vector v1=p.getMajorJobs();
          Vector v2=p.getMinorJobs();

          v1.removeAllElements();
          p.setMajorJobs(v1);
          v2.removeAllElements();
          p.setMinorJobs(v2);
          int personid=Integer.parseInt(p.getID());
          setperson_queue(personid, p);
        }
        savePersonnelVector(person_queue);
}
catch(Exception e){}
    }
    void setperson_queue(int personid, Personnel p1){
      try{
        for(int i=0; i<person_queue.size(); i++){
          Personnel p=(Personnel) person_queue.elementAt(i);
          if(personid==Integer.parseInt(p.getID())){
            person_queue.setElementAt(p1, i);
          }
        }//end for_loop
}
catch(Exception e){}
    }

    void setstep(String stepname, StepContent step1){
      for(int i=0; i<job_pool.size(); i++){
        StepContent step=(StepContent) job_pool.elementAt(i);
        if(step.getStepName().equals(stepname)){
          job_pool.setElementAt(step1, i);
        }
      }
    }
  ////////////////////////////////// END JOB SCHEDULE //////////////////////////////
}
```

```
package Cases;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

////////////////////////////////////////////////////////////////////////
/**
 * CasesTitle : contains all global variables in Cases package.
 * All the titles, directories, or the other names are located in this class
 */
////////////////////////////////////////////////////////////////////////
public interface CasesTitle{

        //Locations of Cases and stakeholder directory
        static final File CASESDIRECTORY = new
File("F:\\nps\\Thesis\\CasesJobSchedule_10_30_99\\cases_le11\\data\\cases
\\");
        static final File STAKEHOLDER = new
File("F:\\nps\\Thesis\\CasesJobSchedule_10_30_99\\cases_le11\\data\\stake
holder\\");

  //Locations of netscape, notepad, winword, excel, and caps
  static final String NETSCAPE = "C:\\Program
Files\\Netscape\\Communicator\\Program\\netscape.exe";
  static final String NOTEPAD = "notepad.exe";
```

```
   static final String WINWORD = "C:\\Program Files\\Microsoft
Office\\Office\\Winword.exe";
   static final String EXCEL = "C:\\Program Files\\Microsoft
Office\\Office\\Excel.exe";
   static final String CAPS = "caps.bat";
   static final String[] EXECUTIONS = {NOTEPAD, WINWORD,
EXCEL, null, NETSCAPE, CAPS};

   //Names of files which Cases will generate
   static final String COMPONENT_CFG = "component.cfg";
   static final String CURRENT_VSN = "current.vsn";
   static final String DEPENDENCY_CFG = "dependency.cfg";
   static final String LOOP_CFG = "loop.cfg";
   static final String STEP_CFG = "step.cfg";

   //Titles of frames under Spider menu
         static final String EDIT_TITLE = "SPIDER-Edit";
         static final String DECOMPOSE_TITLE = "SPIDER-
Decompose";
         static final String COMPONENT_CONTENT_TITLE =
"SPIDER-Component Content";
         static final String STEP_CONTENT_TITLE = "SPIDER-Step
Content";
         static final String TRACE_TITLE = "SPIDER-Trace";

   //title of combo boxes
   static final String PROCESS_TITLE = "Select a Process";
         static final String STEP_TYPE_TITLE = "Select a Step Type";
   static final String COMPONENT_TYPE_TITLE = "Select a Component
Type";
   static final String LINKS_TITLE = "Select a Link Type";
   static final String SKILL_TITLE = "ID : Name : Level";
   static final String[] BUTTONS = {"Add", "Edit", "Delete"};

   //Links file name inside Component Content directory
(COMPONENT_CONTENT_DIR)
   static final String COMPONENT_CONTENT_DIR = "Component
Content";
   static final String[] LINK_FILE_NAMES = {"txt.link", "word.link",
"excel.link", "data.link", "url.link", "caps.link"};

   //two types of variant in EHL merging frame
   static final String[] VARIANT_TYPES = {"Old", "New"};


   //data is using for SkillTable
   static final int SKILL_LEVEL = 4;      //from 0 to 3
   static final int SKILL_ID = 21;        //from 1 to 20
   static final int SECURITY_LEVEL = 6;   //from 0 to 5
   static final int PRIORITY_LEVEL = 6;   //from 0 to 5
   static final String[] SKILL_LIST = {"Unix System","CAPS","TAE
Plus", "C", "C++",
                  "Ada", "Notepad", "MS Word", "MS Excel",
"Rational Rose",
                  "UML", "System Analysis", "System Design",
"Coding", "Testing",
                  "Maintenance", "Organization", "Evaluation",
"Management","Negotiation"};//20 skills name
}
```

```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;
import com.symantec.itools.swing.borders.LineBorder;

////////////////////////////////////////////////////////////////////////
/**
 * ComponentContentFrame : create/edit/delete Component Content directory with all link
 * files for a selected step.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_ComponentContent
 */
////////////////////////////////////////////////////////////////////////
public class ComponentContentFrame extends com.sun.java.swing.JFrame implements CasesTitle, I_ComponentContent
{
   public Vector fnList = new Vector();
   public String pathName = null;
   public String currentComponent = null;

   /**
    * Create ComponentContentFrame
    */
        public ComponentContentFrame()
        {
                // This code is automatically generated by Visual Cafe when you add
                // components to the visual environment. It instantiates and initializes
                // the components. To modify the code, only use code syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("SPIDER-Component Content");
                getContentPane().setLayout(null);
                setSize(650,520);
                setVisible(false);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConstants.RIGHT);
                JLabel1.setText("CAPS Files:");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(30,370,90,20);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConstants.RIGHT);
                JLabel2.setText("Data Files:");
                getContentPane().add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2.setBounds(30,290,90,20);

        JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConstants.RIGHT);
                JLabel3.setText("URLs");
                getContentPane().add(JLabel3);
                JLabel3.setForeground(java.awt.Color.black);
                JLabel3.setBounds(30,330,90,20);

        JLabel4.setHorizontalAlignment(com.sun.java.swing.SwingConstants.RIGHT);
                JLabel4.setText("Text Files:");
                getContentPane().add(JLabel4);
                JLabel4.setForeground(java.awt.Color.black);
                JLabel4.setBounds(30,170,90,20);
                getContentPane().add(textComboBox);
```

```
                textComboBox.setBounds(120,170,500,20);
                getContentPane().add(CAPSComboBox);
                CAPSComboBox.setBounds(120,370,500,20);
                getContentPane().add(URLComboBox);
                URLComboBox.setBounds(120,330,500,20);
                getContentPane().add(saveButton);
                saveButton.setBounds(0,0,0,0);
                getContentPane().add(editButton);
                editButton.setBounds(0,0,0,0);
                getContentPane().add(deleteButton);
                deleteButton.setBounds(0,0,0,0);
                getContentPane().add(addButton);
                addButton.setBounds(0,0,0,0);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("Cancel");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(564,450,73,24);

        JLabel5.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel5.setText("Component Content:");
                getContentPane().add(JLabel5);
                JLabel5.setForeground(java.awt.Color.black);
                JLabel5.setFont(new Font("Dialog", Font.BOLD, 16));
                JLabel5.setBounds(0,7,170,30);
                componentLabel.setText("Component Content:");
                getContentPane().add(componentLabel);
                componentLabel.setForeground(java.awt.Color.black);
                componentLabel.setFont(new Font("Dialog",
Font.BOLD, 15));
                componentLabel.setBounds(175,7,500,30);

        JLabel6.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel6.setText("Available Components:");
                getContentPane().add(JLabel6);
                JLabel6.setForeground(java.awt.Color.black);
                JLabel6.setBounds(10,60,130,20);
                getContentPane().add(componentsComboBox);
                componentsComboBox.setBounds(140,60,500,20);
                getContentPane().add(connectButton);
                connectButton.setBounds(0,0,0,0);
                getContentPane().add(MSWordComboBox);
                MSWordComboBox.setBounds(120,210,500,20);
                getContentPane().add(MSExcelComboBox);
                MSExcelComboBox.setBounds(120,250,500,20);

        JLabel7.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel7.setText("MS Word Files:");
                getContentPane().add(JLabel7);
                JLabel7.setForeground(java.awt.Color.black);
                JLabel7.setBounds(30,210,90,20);

        JLabel8.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel8.setText("MS Excel Files:");
                getContentPane().add(JLabel8);
                JLabel8.setForeground(java.awt.Color.black);
                JLabel8.setBounds(30,250,90,20);
                JPanel1.setBorder(lineBorder1);
                JPanel1.setLayout(null);
                getContentPane().add(JPanel1);
                JPanel1.setBounds(10,120,630,300);
                JPanel1.add(dataComboBox);
                dataComboBox.setBounds(110,168,500,20);

        JLabel9.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel9.setText("Component Content Links");
                JLabel9.setBorder(lineBorder1);
                JPanel1.add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setFont(new Font("Dialog", Font.BOLD, 15));
```

```
                JLabel9.setBounds(236,0,190,24);
                getContentPane().add(JPanel1);
                JPanel1.setFont(new Font("Dialog", Font.BOLD, 12));
                //$$ lineBorder1.move(0,521);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                addButton.addActionListener(lSymAction);
                editButton.addActionListener(lSymAction);
                deleteButton.addActionListener(lSymAction);
                saveButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                SymItem iSymItem = new SymItem();
                componentsComboBox.addItemListener(iSymItem);
                //}}
        }

    /**
     * Constructor for CasesFrame to launch ComponentContentFrame
     */
    public ComponentContentFrame( String pathName, String stepName,
Vector itemVector, Vector fnList ){
        this();
        //Add Save button
                saveButton.setText("Save");
                saveButton.setActionCommand("Save");
                getContentPane().add(saveButton);
                saveButton.setBounds(492,450,70,24);

                //Add Edit button
                editButton.setText("Edit");
                editButton.setActionCommand("Edit");
                getContentPane().add(editButton);
                editButton.setBounds(81,450,70,24);

                //Add Delete button
                deleteButton.setText("Delete");
                deleteButton.setActionCommand("Delete");
                getContentPane().add(deleteButton);
                deleteButton.setBounds(152,450,70,24);

                //Add Add button
                addButton.setText("Add");
                addButton.setActionCommand("Add");
                getContentPane().add(addButton);
                addButton.setBounds(10,450,70,24);

        this.fnList = fnList;
        this.componentLabel.setText(stepName);
        this.pathName = pathName;
        this.componentsComboBox.addItem(COMPONENT_TYPE_TITLE);
        for( int i=0; i<itemVector.size(); i++ ){
          this.componentsComboBox.addItem( itemVector.elementAt(i) );
        }
    }

    /**
     * Constructor for TraceFrame to launch ComponentContentFrame
     */
    public ComponentContentFrame( String pathName, String stepName,
String currentItem, Vector fnList ){
        this();
        this.fnList = fnList;
        this.componentLabel.setText(stepName);
        this.pathName = pathName;
        this.componentsComboBox.addItem(currentItem);
        this.componentsComboBox.setSelectedItem(currentItem);
        this.componentsComboBox.setEnabled(false);

        //Connect button is used in trace panel
```

```
                connectButton.setText("Connect");
                getContentPane().add(connectButton);
                connectButton.setBounds(517,380,81,24);
}

        public ComponentContentFrame(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new ComponentContentFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel4 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox textComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox CAPSComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox URLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton saveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton editButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton deleteButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton addButton = new
com.sun.java.swing.JButton();
```

```
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel5 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel componentLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel6 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox componentsComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton connectButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JComboBox MSWordComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox MSExcelComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel7 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel8 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JPanel JPanel1 = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JComboBox dataComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.symantec.itools.swing.borders.LineBorder lineBorder1 = new
com.symantec.itools.swing.borders.LineBorder();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == addButton)
                                addButton_actionPerformed(event);
                        else if (object == editButton)
                                editButton_actionPerformed(event);
                        else if (object == deleteButton)
                                deleteButton_actionPerformed(event);
                        else if (object == saveButton)
                                saveButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                }
        }

  /**
    * Launch ConnectionLinksFrame
    * Adding more connection links to a componentContent
    */
  public void addButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          String s =
(String)this.componentsComboBox.getSelectedItem();
     (new ConnectionLinksFrame(this, s, BUTTONS[0])).setVisible( true );
        }

  /**
    * Launch ConnectionLinksFrame
    * Editting existing connection links in a componentContent
    */
        public void
editButton_actionPerformed(java.awt.event.ActionEvent event)
        {
```

```
            String s =
(String)this.componentsComboBox.getSelectedItem();
      (new ConnectionLinksFrame(this, s, BUTTONS[1])).setVisible( true );
          }

   /**
    * Launch ConnectionLinksFrame
    * Delete connection links in a componentContent
    */
          public void
deleteButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            String s =
(String)this.componentsComboBox.getSelectedItem();
      (new ConnectionLinksFrame(this, s, BUTTONS[2])).setVisible( true );
          }

   /**
    * Save all link files under Component Content directory of
    * currentComponent
    */
          public void
saveButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
        saveLinkFile( LINK_FILE_NAMES[i] );
      }
            clearComboBoxes();
            setVisible(false);
            dispose();
          }

   /**
    * Clear all items in every combo box
    * Exit ComponentContentFrame
    */
          public void
cancelButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            clearComboBoxes();
            setVisible(false);
            dispose();
          }

          class SymItem implements java.awt.event.ItemListener
          {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                      Object object = event.getSource();
                      if (object == componentsComboBox)

          componentsComboBox_itemStateChanged(event);
                }
          }

   /**
    * Select an item in the combo box
    */
   public void
componentsComboBox_itemStateChanged(java.awt.event.ItemEvent event)
          {
            String selectedItem = null;
      if( event.getStateChange() == event.SELECTED ){
        selectedItem = (String)event.getItem();
        if( !selectedItem.equals(COMPONENT_TYPE_TITLE) ){
          clearComboBoxes();
          setButtonEnabled( true );
          this.currentComponent = (String)checkInput(selectedItem);
          if( !this.currentComponent.equals("") ){
            File f = (File)searchPath(this.currentComponent);
            if( !f.exists() ){
```

```
            JOptionPane.showMessageDialog(this, selectedItem+" does
not exist!", "Alert Message",
                                    JOptionPane.ERROR_MESSAGE);
          }
          else if( f.exists() ){
            String[] list = f.list();
            for( int j=0; j<list.length; j++ ){
              String s = (String)list[j];
              File aFile = null;
              if( s.equals(COMPONENT_CONTENT_DIR) ){
                aFile = new File(f, s);
                if( !aFile.isDirectory() ){
                  aFile.mkdir();
                  j= list.length;
                  setButtonEnabled( false );
                }
                else if( aFile.isDirectory() ){
                  j = list.length;
                  setButtonEnabled( true );
                  list = aFile.list();
                  for( int i=0; i<list.length; i++ ){
                    searchFiles(aFile, (String)list[i]);
                  }
                }
              }
              else if( !s.equals(COMPONENT_CONTENT_DIR) ){
                aFile = new File(f, COMPONENT_CONTENT_DIR);
                aFile.mkdir();
                setButtonEnabled( true );
              }
            }
          }
        }
        else if( selectedItem.equals(COMPONENT_TYPE_TITLE) ){
          setButtonEnabled( false );
        }
      }
    }

    /**
     * Add/Delete/Edit/Save button only active when a step is selected
     */
        public void setButtonEnabled( boolean flag ){
          addButton.setEnabled( flag );
          editButton.setEnabled( flag );
          deleteButton.setEnabled( flag );
          saveButton.setEnabled( flag );
        }

    /**
     * Short cut to print the output
     * @param string : the output string
     */
        void debug( String s ){
          System.out.println(s);
        }

        /**
         * Search and get the info from link files
         *
         * @param aFile : a link file
         * @param fileType : a name of link file
         */
        public void searchFiles(File aFile, String fileType){
      File f = new File( aFile, fileType);
      try{
        BufferedReader br = null;
        String item = null;
        if( fileType.equals(LINK_FILE_NAMES[0]) ){
          br = new BufferedReader( new FileReader(f));
          if( br != null ){
            while( (item = br.readLine()) != null ){
              this.textComboBox.addItem(item);
```

```
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[1]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.MSWordComboBox.addItem(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[2]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.MSExcelComboBox.addItem(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[3]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.dataComboBox.addItem(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[4]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.URLComboBox.addItem(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[5]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.CAPSComboBox.addItem(item);
        }
      }
    }
  }
  catch( IOException io ){
    debug("IOException: "+io);
  }
    }

/**
 * Form an absolute path for a selected component
 *
 * @param s : a selected component name
 * @return File which is formed by a string s
 */
public File searchPath( String s){
  String thePath = null;

  s = "s-"+s;
  //To convert the string of selected item into the file path
  for( int i=0; i<this.fnList.size(); i++ ){
    String fn = (String)this.fnList.elementAt(i);
    String sub = s.substring(0,fn.length());

    if( sub.equals(fn)){
      i = this.fnList.size();
      thePath= sub+"\\";
      String sub2 = s.substring(fn.length());

      char[] ca = (char[])sub2.toCharArray();
      int result = 0;
      for( int j=0; j<ca.length; j++ ){
        String s3 = ca[j]+"";
        result = s3.compareTo("-");
```

```
            if( result == 0 ){
               j=ca.length;
               StringTokenizer st = new StringTokenizer(sub2,"-");
                     String before_ = st.nextToken("-");
                     thePath = thePath+before_;
                     String _after = st.nextToken("-");
                     st = new StringTokenizer(_after,".");
                     while(st.hasMoreTokens()){
                        thePath = thePath+"\\"+st.nextToken();
                     }
            }
            else if( (result > 0) && (j==ca.length - 1) ){
               thePath = thePath+sub2;
            }
         }
      }
   }
   File f = new File(this.pathName,thePath);
   return f;
}

/**
 * Set text file names in textComboBox
 *
 * @param v : a vector of links
 */
public void setTextLink(Vector v){
   this.textComboBox.removeAllItems();
   for( int i=0; i<v.size(); i++ ){
      this.textComboBox.addItem(v.elementAt(i));
   }
}

/**
 * Set word file names in MSWordComboBox
 *
 * @param v : a vector of links
 */
public void setWordLink(Vector v){
   this.MSWordComboBox.removeAllItems();
   for( int i=0; i<v.size(); i++ ){
      this.MSWordComboBox.addItem(v.elementAt(i));
   }
}

/**
 * Set excel file names in MSExcelComboBox
 *
 * @param v : a vector of links
 */
public void setExcelLink(Vector v){
   this.MSExcelComboBox.removeAllItems();
   for( int i=0; i<v.size(); i++ ){
      this.MSExcelComboBox.addItem(v.elementAt(i));
   }
}

/**
 * Set personnel object names in dataComboBox
 *
 * @param v : a vector of links
 */
public void setDataLink(Vector v){
   this.dataComboBox.removeAllItems();
   for( int i=0; i<v.size(); i++ ){
      this.dataComboBox.addItem(v.elementAt(i));
   }
}

/**
 * Set URL in URLComboBox
 *
 * @param v : a vector of links
 */
```

```
  public void setURLLink(Vector v){
    this.URLComboBox.removeAllItems();
    for( int i=0; i<v.size(); i++ ){
      this.URLComboBox.addItem(v.elementAt(i));
    }
  }

  /**
   * Set caps file names in CAPSComboBox
   *
   * @param v : a vector of links
   */
  public void setCAPSLink(Vector v){
    this.CAPSComboBox.removeAllItems();
    for( int i=0; i<v.size(); i++ ){
      this.CAPSComboBox.addItem(v.elementAt(i));
    }
  }

  /**
   * Save all link files
   *
   * @param fileType : a type of file, e.g., txt.link, data.link, ...
   */
  public void saveLinkFile( String fileType){
    Vector items = new Vector();
    JComboBox comboBox = (JComboBox)findComboBox( fileType );
    for( int i=0; i<comboBox.getItemCount(); i++ ){
      items.addElement(comboBox.getItemAt(i));
    }

    File f = (File)searchPath(this.currentComponent);
    f = new
File(f.getAbsolutePath()+"\\"+COMPONENT_CONTENT_DIR+"\\"+fileT
ype);
    try{
      FileWriter fw = new FileWriter(f);
      BufferedWriter bw = new BufferedWriter( fw);
      if( bw != null ){
        for( int i=0; i<comboBox.getItemCount(); i++ ){
          bw.write((String)comboBox.getItemAt(i)+"\n");
        }
      }
      bw.flush();
      bw.close();
      fw.close();
    }
    catch( IOException io ){
      debug("IOException: "+io);
    }
  }

  /**
   * Search a combobox matchs with fileType
   *
   * @param fileType : a type of file, e.g., txt.link, data.link, ...
   * @return JComboBox : a combo box of fileType
   */
  public JComboBox findComboBox( String fileType ){
    JComboBox comboBox = null;
    if( fileType.equals(LINK_FILE_NAMES[0]) ){
      comboBox = this.textComboBox;
    }
    else if( fileType.equals(LINK_FILE_NAMES[1]) ){
      comboBox = this.MSWordComboBox;
    }
    else if( fileType.equals(LINK_FILE_NAMES[2]) ){
      comboBox = this.MSExcelComboBox;
    }
    else if( fileType.equals(LINK_FILE_NAMES[3]) ){
      comboBox = this.dataComboBox;
    }
    else if( fileType.equals(LINK_FILE_NAMES[4]) ){
      comboBox = this.URLComboBox;
```

```
        }
        else if( fileType.equals(LINK_FILE_NAMES[5]) ){
            comboBox = this.CAPSComboBox;
        }
        return comboBox;
    }

    /**
     * Refresh all comboboxes
     */
    public void clearComboBoxes(){
        textComboBox.removeAllItems();
        MSWordComboBox.removeAllItems();
        MSExcelComboBox.removeAllItems();
        dataComboBox.removeAllItems();
        URLComboBox.removeAllItems();
        CAPSComboBox.removeAllItems();
    }

    /**
     * Check the selected component is valid or not
     *
     * @return String : null/valid component name
     * @param selectedItem : a selected component in
componentComboBox
     */
    public String checkInput(String selectedItem ){
        if( selectedItem != null ){
            int j = selectedItem.indexOf("-");
            if( j>0){
                    String sub2 = (selectedItem.substring(j+1)).trim();
                if( !sub2.equals(this.componentLabel.getText()) ){
                        char c = (char)sub2.charAt(0);
                        boolean isLetter = (new Character(c)).isLetter(c);
                        if( isLetter ){
                            JOptionPane.showMessageDialog(this, selectedItem+"
is invalid component!",
                                                    "Error
Message",JOptionPane.ERROR_MESSAGE);
                            return null;
                            }
                        }
                        else{
                            return this.componentLabel.getText();
                        }
                    }
                }
            return selectedItem;
                }
    }
```

```
package Cases;

import java.io.Serializable;

////////////////////////////////////////////////////////////////////////
/**
 * ComponentType : Create a Component Type object and save it in
component.cfg file
 */
////////////////////////////////////////////////////////////////////////
public class ComponentType implements Serializable{

    /**
     * componentID : component type ID
     */
    private String componentID;

    /**
     * componentName : component type name
```

```
    */
   private String componentName;

   /**
    * componentDescription : component type description
    */
   private String componentDescription;

   /**
    * This ComponentType constructor is used to create a ComponentType
object
    */
   public ComponentType( String componentID, String componentName,
String componentDescription ){
      this.componentID = componentID;
      this.componentName = componentName;
      this.componentDescription = componentDescription;
   }

   public ComponentType(){}
   /**
    * @return componentID
    */
   public String getComponentID(){
      return this.componentID;
   }

   /**
    * @return componentName
    */
   public String getComponentName(){
      return this.componentName;
   }

   /**
    * @return componentDescription
    */
   public String getComponentDescription(){
      return this.componentDescription;
   }
}
```



```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * ConnectionLinksFrame : Create/Edit/Delete links of a selected
component
 * Launched by ComponentContentFrame's buttons(Add, Edit, or Delete)
 *
 * Implement CasesTitle where stores all global variables of Cases package
 */
////////////////////////////////////////////////////////////////////////
public class ConnectionLinksFrame extends com.sun.java.swing.JFrame
                                          implements CasesTitle
{
   /**
    * fd: instantiate an open fileDialog
    */
   FileDialog fd = new FileDialog(this, "Open",FileDialog.LOAD);

   /**
    * update : to check that all the changes is update yet
    */
   boolean update = true;
```

```
/**
 * compContentFrame : get the current ComponentContentFrame
 */
ComponentContentFrame compContentFrame = null;

/**
 * listModel : to monitor item list
 */
DefaultListModel listModel = new DefaultListModel();

/**
 * selectedItem : current link type
 */
private String selectedItem = null;

/**
 * currentPosition : position of selected item in the itemList
 */
private int currentPosition = 0;

/**
 * button : name of button of ComponentContentFrame to launch
ConnectionLinksFrame
 */
String button = null;

/**
 * storedVector : an array of  vector of all links
 */
Vector[] storedVector = {new Vector(), new Vector(), new Vector(), new
Vector(), new Vector(), new Vector()};


/**
 * Build ConnectionLinksFrame
 */
```

```
public ConnectionLinksFrame()
{
        // This code is automatically generated by Visual Cafe
when you add
        // components to the visual environment. It instantiates
and initializes
        // the components. To modify the code, only use code
syntax that matches
        // what Visual Cafe can generate, or Visual Cafe may be
unable to back
        // parse your Java file into its visual environment.
        //{{INIT_CONTROLS
        setTitle("Component Content Editor");
        getContentPane().setLayout(null);
        setSize(500,450);
        setVisible(false);

    titleLabel.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.CENTER);
        titleLabel.setText("jlabel");
        getContentPane().add(titleLabel);
        titleLabel.setForeground(java.awt.Color.black);
        titleLabel.setFont(new Font("Dialog", Font.BOLD, 18));
        titleLabel.setBounds(25,5,450,30);
        exitButton.setText("Exit");
        exitButton.setActionCommand("Cancel");
        getContentPane().add(exitButton);
        exitButton.setBounds(260,400,75,22);
        itemScrollPane.setOpaque(true);
        getContentPane().add(itemScrollPane);
        itemScrollPane.setBounds(15,165,470,200);
        itemScrollPane.getViewport().add(itemList);
        itemList.setBounds(0,0,467,197);
        updateButton.setText("Update");
        updateButton.setActionCommand("OK");
        getContentPane().add(updateButton);
        updateButton.setBounds(164,400,75,22);
```

```
                getContentPane().add(connectionComboBox);
                connectionComboBox.setBounds(115,60,270,22);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel1.setText("Connection Links");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(15,140,470,22);
                getContentPane().add(tempTextField);
                tempTextField.setBounds(12,100,321,22);
                tempButton.setText("jbutton");
                tempButton.setActionCommand("jbutton");
                getContentPane().add(tempButton);
                tempButton.setBounds(333,100,75,22);
                browseButton.setText("Browse");
                browseButton.setActionCommand("Browse");
                getContentPane().add(browseButton);
                browseButton.setBounds(408,100,78,22);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymItem lSymItem = new SymItem();
                connectionComboBox.addItemListener(lSymItem);
                SymAction lSymAction = new SymAction();
                tempButton.addActionListener(lSymAction);
                updateButton.addActionListener(lSymAction);
                exitButton.addActionListener(lSymAction);
                SymMouse aSymMouse = new SymMouse();
                itemList.addMouseListener(aSymMouse);
                browseButton.addActionListener(lSymAction);
                //}}

        /**
```

```
                * set default model for itemList
                */
                itemList.setModel(listModel);

                /**
                * create a combobox with all type of links
                */
                for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
            connectionComboBox.addItem(LINK_FILE_NAMES[i]);
        }
    }

    /**
     * Constructor is launched by buttons from ComponentContentFrame
     */
    public ConnectionLinksFrame(ComponentContentFrame
compContentFrame, String title, String button){
        this();
        this.compContentFrame = compContentFrame;
        this.button = button;
        this.titleLabel.setText(title);
        this.tempButton.setText(button);
        setListModel();
    }

        public ConnectionLinksFrame(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }
```

```
	static public void main(String args[])
	{
		(new ConnectionLinksFrame()).setVisible(true);
	}

	public void addNotify()
	{
		// Record the size of the window prior to calling parents
addNotify.
		Dimension size = getSize();

		super.addNotify();

		if (frameSizeAdjusted)
			return;
		frameSizeAdjusted = true;

		// Adjust size of frame according to the insets and menu
bar
		Insets insets = getInsets();
		com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
		int menuBarHeight = 0;
		if (menuBar != null)
			menuBarHeight =
menuBar.getPreferredSize().height;
		setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
	}

	// Used by addNotify
	boolean frameSizeAdjusted = false;

	//{{DECLARE_CONTROLS
	com.sun.java.swing.JLabel titleLabel = new
com.sun.java.swing.JLabel();
		com.sun.java.swing.JButton exitButton = new
com.sun.java.swing.JButton();
		com.sun.java.swing.JScrollPane itemScrollPane = new
com.sun.java.swing.JScrollPane();
		com.sun.java.swing.JList itemList = new
com.sun.java.swing.JList();
		com.sun.java.swing.JButton updateButton = new
com.sun.java.swing.JButton();
		com.sun.java.swing.JComboBox connectionComboBox = new
com.sun.java.swing.JComboBox();
		com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
		com.sun.java.swing.JTextField tempTextField = new
com.sun.java.swing.JTextField();
		com.sun.java.swing.JButton tempButton = new
com.sun.java.swing.JButton();
		com.sun.java.swing.JButton browseButton = new
com.sun.java.swing.JButton();
	//}}

	//{{DECLARE_MENUS
	//}}


	class SymItem implements java.awt.event.ItemListener
	{
		public void itemStateChanged(java.awt.event.ItemEvent
event)
		{
			Object object = event.getSource();
			if (object == connectionComboBox)

	connectionComboBox_itemStateChanged(event);
		}
	}

/**
```

```
    * Monitor what link type is selected
    */
        void
connectionComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == event.SELECTED ){
          listModel.removeAllElements();
            this.tempTextField.setText("");
            selectedItem = (String)event.getItem();
            for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
         if( selectedItem.equals(LINK_FILE_NAMES[i]) ){
            for( int j=0; j<storedVector[i].size(); j++ ){
              listModel.addElement(storedVector[i].elementAt(j));
            }
            i= LINK_FILE_NAMES.length;
          }
        }
         }
         }

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == tempButton)
                                tempButton_actionPerformed(event);
                        else if (object == updateButton)
                                updateButton_actionPerformed(event);
                        else if (object == exitButton)
                                exitButton_actionPerformed(event);
                        else if (object == browseButton)
                                browseButton_actionPerformed(event);
                }
        }
```



```
        void tempButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      String s = tempTextField.getText();
      if( !s.equals("") ){
        searchButton(s);
              this.tempTextField.setText("");
        update = false;
      }
      else{
        update = true;
      }
        }

   /**
    * Update all link comboboxes in ComponentContentFrame
    */
        void updateButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          int index = (int)connectionComboBox.getSelectedIndex();
          searchLink(index);
          String s = (String)tempTextField.getText();
          if( !s.equals("") ){
            searchButton(s);
          }
          update = true;
        }

   /**
    * Confirmination dialog to give a user one more chance to update all
    * the change, and exit ConnectionLinksFrame.
    */
        void exitButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      if( !update ){
```

```
            Object[] options = { "OK", "CANCEL" };
        int i = JOptionPane.showOptionDialog(this, "You did not update the
last change yet.  Would you like to update?",
        "Warning", JOptionPane.DEFAULT_OPTION,
JOptionPane.WARNING_MESSAGE, null, options, options[0]);
        if( i==0 ){
           updateButton_actionPerformed(event);
        }
      }
      setVisible( false );
      dispose();
          }

   /**
    * Connect to FileDialog to let user to select a file insteads of typing, and
return the
    * entire path of the file to the textfield
    */
          void browseButton_actionPerformed(java.awt.event.ActionEvent
event)
          {
      fd.show();
      String fileName = fd.getFile();
      if( fileName != null ){
        tempTextField.setText(fd.getDirectory()+fileName);
      }
          }


          class SymMouse extends java.awt.event.MouseAdapter
          {
                    public void mouseClicked(java.awt.event.MouseEvent
event)
                    {
                              Object object = event.getSource();
                              if (object == itemList)
                                        itemList_mouseClicked(event);
                    }
          }

    /**
     * Monitor which item is selected
     */
          void itemList_mouseClicked(java.awt.event.MouseEvent event)
          {
            if( event.getClickCount() > 0) {
              String s = (String)this.itemList.getSelectedValue();
              this.tempTextField.setText(s);
              this.currentPosition = (int)this.itemList.getSelectedIndex();
            }
    }

    /**
     * Search which button (Add/Delete/Edit) from
ComponentContentFrame to
     * launch this frame.
     *
     * @param s : button title which is passed from
ComponentContentFrame
     */
          void searchButton(String s){
            for( int i=0; i<BUTTONS.length; i++ ){
              if( this.button.equals(BUTTONS[i]) ){
                searchVector(s,i);
                i=BUTTONS.length;
              }
            }
          }

    /**
     * Search which comboBox in ComponentContentFrame needs to update
     *
     * @param index : an index of selected item in connectionComboBox
     */
```

```
        void searchLink(int index){
    if( index == 0 ){
            this.compContentFrame.setTextLink(storedVector[0]);
          }
    else if( index == 1 ){
            this.compContentFrame.setWordLink(storedVector[1]);
          }
    else if( index == 2 ){
            this.compContentFrame.setExcelLink(storedVector[2]);
          }
    else if( index == 3 ){
            this.compContentFrame.setDataLink(storedVector[3]);
          }
    else if( index == 4 ){
            this.compContentFrame.setURLLink(storedVector[4]);
          }
    else if( index == 5 ){
            this.compContentFrame.setCAPSLink(storedVector[5]);
          }
        }

  /**
   * Set the content of a selected link file into the list
   */
  void setListModel(){
    if( this.compContentFrame != null ){
      for( int j=0; j<LINK_FILE_NAMES.length; j++ ){
        JComboBox cb =
(JComboBox)this.compContentFrame.findComboBox(LINK_FILE_NAME
S[j]);
        int c =cb.getItemCount();
        for( int i=0; i<c; i++ ){
          storedVector[j].addElement(cb.getItemAt(i));
        }
      }
      for( int j=0; j<storedVector[0].size(); j++ ){
        listModel.addElement(storedVector[0].elementAt(j));
      }
    }
  }

  /**
   * Search which vector is respective with the selected link type
   *
   * @param s : a link type
   * @param index : the index of selected link type and it use to match
with
   * the element in array vector, storedVector.
   */
  void searchVector(String s, int index){
    for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
      if( selectedItem.equals(LINK_FILE_NAMES[i]) ){
        if( index==0 ){
          if( !listModel.contains(s) ){
            listModel.addElement(s);
            storedVector[i].addElement(s);
              }
              else{
                JOptionPane.showMessageDialog(this, s+" is already
in the list!",
                                              "Warning Message",
JOptionPane.ERROR_MESSAGE);
              }
        }
        else if( index==1 ){
          listModel.removeElementAt(this.currentPosition);
          listModel.insertElementAt(s,this.currentPosition);
          storedVector[i].removeElementAt(this.currentPosition);
          storedVector[i].insertElementAt(s,this.currentPosition);
        }
        else if( index==2 ){
          if( listModel.contains(s) ){
            listModel.removeElementAt(this.currentPosition);
            storedVector[i].removeElementAt(this.currentPosition);
```

```
                    }
                    else{
                      JOptionPane.showMessageDialog(this, s+" is not in
the list!",
                                                    "Warning Message",
JOptionPane.ERROR_MESSAGE);
                    }
            }
            i= LINK_FILE_NAMES.length;
          }
        }
    }
}
```



```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * DecomposeListDialog : a dialog stores all decomposed components of a
selected step.
 * It is launched by decomposeComboBox of TraceFrame
 * It has 2 lists, components and decomposed components
 */
////////////////////////////////////////////////////////////////////////
public class DecomposeListDialog extends com.sun.java.swing.JDialog
{
  /**
   * Parent frame of this dialog
   */
  TraceFrame tf = null;

  /**
   * Monitor all components of the selected step
   */
  DefaultListModel componentListModel = new DefaultListModel();

  /**
   * Monitor all decomposed components of the selected component
   */
  DefaultListModel decomposeListModel = new DefaultListModel();

  /**
   * Build DecomposeListDialog
   */
      public DecomposeListDialog(JFrame parent)
      {
            super(parent);

            // This code is automatically generated by Visual Cafe
when you add
            // components to the visual environment. It instantiates
and initializes
            // the components. To modify the code, only use code
syntax that matches
            // what Visual Cafe can generate, or Visual Cafe may be
unable to back
            // parse your Java file into its visual environment.
            //{{INIT_CONTROLS
            setModal(true);
            getContentPane().setLayout(null);
            setSize(585,400);
            setVisible(false);
            componentScrollPane.setOpaque(true);
            getContentPane().add(componentScrollPane);
            componentScrollPane.setBounds(15,90,270,230);
```

```
                componentScrollPane.getViewport().add(componentList);
                componentList.setBounds(0,0,267,227);
                OKButton.setText("OK");
                OKButton.setActionCommand("OK");
                getContentPane().add(OKButton);
                OKButton.setBounds(207,340,75,22);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("Cancel");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(303,340,75,22);

        titleLabel.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.CENTER);
                titleLabel.setText("jlabel");
                getContentPane().add(titleLabel);
                titleLabel.setForeground(java.awt.Color.black);
                titleLabel.setFont(new Font("Dialog", Font.BOLD, 18));
                titleLabel.setBounds(42,5,500,30);
                decomposeScrollPane.setOpaque(true);
                getContentPane().add(decomposeScrollPane);
                decomposeScrollPane.setBounds(300,90,270,230);

        decomposeScrollPane.getViewport().add(decomposeList);
                decomposeList.setBounds(0,0,267,227);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);

        JLabel1.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel1.setText("Component");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(15,68,270,22);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
        JLabel2.setVerticalAlignment(com.sun.java.swing.SwingConstant
s.BOTTOM);
                JLabel2.setText("Decompose");
                getContentPane().add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2.setBounds(300,68,270,22);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                SymMouse aSymMouse = new SymMouse();
                componentList.addMouseListener(aSymMouse);
                //}}

                /**
                 * setModel for componentList and decomposeList
                 */
                componentList.setModel(componentListModel);
                decomposeList.setModel(decomposeListModel);
        }

  /**
   * TraceFrame use this constructor to launch DecomposeListDialog
   */
        public DecomposeListDialog(TraceFrame tf, String title, Vector
itemVector)
        {
                this((JFrame)tf);
                this.tf = tf;
                setTitle(title);
                this.titleLabel.setText(title);
                for( int i=0; i<itemVector.size(); i++ ){

componentListModel.addElement(itemVector.elementAt(i));
```

```
            }
        }

        public void setVisible(boolean b)
        {
            if (b)
              setLocation(50, 50);
            super.setVisible(b);
        }


        public void addNotify()
        {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                return;
            frameSizeAdjusted = true;

            // Adjust size of frame according to the insets
            Insets insets = getInsets();
            setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JScrollPane componentScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JList componentList = new
com.sun.java.swing.JList();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel titleLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane decomposeScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JList decomposeList = new
com.sun.java.swing.JList();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
            public void actionPerformed(java.awt.event.ActionEvent
event)
            {
                Object object = event.getSource();
                if (object == OKButton)
                    OKButton_actionPerformed(event);
                else if (object == cancelButton)
                    cancelButton_actionPerformed(event);
            }
        }

    /**
     * Return a selected decompose step to TraceFrame
     */
        void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
```

```
            String selectedItem =
(String)this.decomposeList.getSelectedValue();
            if( selectedItem != null ){
        if( this.tf != null ){
          String s = this.tf.convertToThePath(selectedItem);
          if( s!= null ){
            this.tf.setSelectedItem(s, selectedItem);
            setVisible(false);
            dispose();
          }
        }
      }
      else{
        setVisible(false);
        dispose();
      }
          }

  /**
   * Exit DecomposeListDialog
   */
        void cancelButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      setVisible(false);
      dispose();
        }

        class SymMouse extends java.awt.event.MouseAdapter
        {
                public void mouseReleased(java.awt.event.MouseEvent
event)
                {
                        Object object = event.getSource();
                        if (object == componentList)
                                componentList_mouseReleased(event);
                }
        }

  /**
   * View all decomposed components of the selected component
   */
        void componentList_mouseReleased(java.awt.event.MouseEvent
event)
        {
      if(event.getClickCount() > 0){
        decomposeListModel.removeAllElements();
        if( this.tf != null ){
          String currentComponent =
this.tf.checkSelection((String)this.componentList.getSelectedValue());
          if( !currentComponent.equals("") ){
            String currentPath =
this.tf.convertToThePath(currentComponent);
            if( (currentPath != null) && (!currentPath.equals("")) ){
              File f = new File(currentPath);
              if( f.exists() ){
                Vector decomposedSteps = new Vector();
                decomposedSteps =
(Vector)findSteps(currentComponent,new File(currentPath));
                if( (decomposedSteps != null) &&
(decomposedSteps.size() > 0) ){
                                for( int i=0; i<decomposedSteps.size(); i++
){

decomposeListModel.addElement(decomposedSteps.elementAt(i));
                                }
                }
                else{
                  JOptionPane.showMessageDialog(this, currentPath+"
step is an atomic component.  Please reselect it!",
                                "Warning
Message",JOptionPane.ERROR_MESSAGE);
                  return;
                }
```

```
                }
            }
        }
    }
}
    }

    /**
     * Find all decomposed components of the selected component
     *
     * @param temp : a string of selected component name
     * @param f : a file with a name temp
     * @return decomposes vector contains all decomposed
components of the component temp
     */
  Vector findSteps(String temp, File f){
    Vector decomposes = new Vector();
    String[] list = (String[])f.list();
    for(int i=0; i<list.length; i++ ){
      File f2 = new File(f, list[i]);
      if( f2.isDirectory() ){
        try{
          int num = Integer.parseInt(f2.getName());
          int index = temp.indexOf("-");
          String temp2 = null;
          if( index > 0 ){
            temp2 = temp+"."+f2.getName();
          }
          else{
            temp2 = temp+"-"+f2.getName();
          }
          decomposes.addElement(temp2);
          findSteps(temp2,f2);
        }
        catch(Exception e){}
      }
    }
    return decomposes;
  }
}
```



```
package Cases;

import java.awt.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * DeleteDialog : a dialog uses to delete a personnel object and
 * a project
 */
////////////////////////////////////////////////////////////////////////
public class DeleteDialog extends com.sun.java.swing.JDialog
{
  /**
   * fileDialog : use to get a personnel objects under stakeholder directory
   */
  FileDialog fileDialog = null;

  /**
   * deleteProject : a flag to define a purpose of using this dialog
   */
  boolean deleteProject = false;

  /**
   * casesFrame : a parent frame launch this dialog from two menu items,
   *          Project --> Delete Project and
   *          Tools --> Personnel --> Delete
```

```
    */
    CasesFrame casesFrame = null;

    /**
     * casesDirectory : the entire path of current project, eg.
F:\Cases\projectName\
     */
    public String casesDirectory = null;

    /**
     *
     */
        public DeleteDialog(Frame parent)
        {
                super(parent);

                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Delete  Project");
                setModal(true);
                getContentPane().setLayout(null);
                setSize(300,130);
                setVisible(false);
                OKButton.setText("OK");
                OKButton.setActionCommand("OK");
                getContentPane().add(OKButton);
                OKButton.setBounds(74,70,73,20);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("Cancel");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(152,70,73,20);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                browseButton.addActionListener(lSymAction);
                SymItem lSymItem = new SymItem();
                projectComboBox.addItemListener(lSymItem);
                //}}
        }

        public DeleteDialog()
        {
                this((Frame)null);
        }

/**
 * CasesFrame uses this constructor to launch this dialog
 *
 * @param casesFrame : current CasesFrame
 * @param title : "Delete Project" or "Delete Personnel Data"
 */
        public DeleteDialog(CasesFrame casesFrame, String title){
            this(title);
            this.casesFrame = casesFrame;
            if( title.equals("Delete Project") ){
                    getContentPane().add(projectComboBox);
                    projectComboBox.setBounds(161,30,128,20);
                    projectTextField.setEditable(false);
                    getContentPane().add(projectTextField);
                    projectTextField.setBackground(java.awt.Color.white);
                    projectTextField.setBounds(10,30,150,20);
```

```
            casesDirectory =
this.casesFrame.CASESDIRECTORY.getAbsolutePath()+"\\";
            projectTextField.setText(casesDirectory);

            String[] list = this.casesFrame.CASESDIRECTORY.list();

            if( list.length > 0 ){
              for(int i=0; i<list.length; i++ ){
                File aFile = new File(casesDirectory, list[i]);
                if( aFile.isDirectory() ){
                  projectComboBox.addItem(list[i]);
                }
              }
            }
            deleteProject = true;
          }
          else if( title.equals("Delete Personnel Data") ){
        fileDialog = new FileDialog(this.casesFrame, title);
            getContentPane().add(filenameTextField);
            filenameTextField.setBounds(10,30,200,20);
            browseButton.setText("Browse");
            browseButton.setActionCommand("Browse");
            getContentPane().add(browseButton);
            browseButton.setBounds(211,30,78,20);

            deleteProject = false;
          }
        }

        public DeleteDialog(String sTitle)
        {
            this();
            setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
            if (b)
                setLocation(50, 50);
            super.setVisible(b);
        }

        static public void main(String args[])
        {
            (new DeleteDialog()).setVisible(true);
        }

        public void addNotify()
        {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                return;
            frameSizeAdjusted = true;

            // Adjust size of frame according to the insets
            Insets insets = getInsets();
            setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JTextField filenameTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
```

```
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton browseButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JComboBox projectComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextField projectTextField = new
com.sun.java.swing.JTextField();
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                        else if (object == browseButton)
                                browseButton_actionPerformed(event);
                }
        }

  /**
   * To check deleteProject or deletePersonnel before delete it
   */
        void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    if( deleteProject ){
            File aFile = new File(projectTextField.getText());
      removeAll(aFile);
    }
    else{
      File aFile = new File(filenameTextField.getText());
      aFile.delete();
    }
    setVisible(false);
    dispose();
        }

  /**
   * Exit DeleteDialog
   */
        void cancelButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    setVisible(false);
    dispose();
        }

  /**
   * Show fileDialog with current directory is stakeholder and list all
personnel file
   */
        void browseButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    fileDialog.setMode(FileDialog.LOAD);

fileDialog.setDirectory(this.casesFrame.STAKEHOLDER.getAbsolutePath
());
    fileDialog.show();
    if( fileDialog.getFile() != null ){

filenameTextField.setText(fileDialog.getDirectory()+fileDialog.getFile());
    }
    else{
      filenameTextField.setText("");
    }
  }
```

```
        class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == projectComboBox)

        projectComboBox_itemStateChanged(event);
                }
        }

  /**
    * projectComboBox contains all project names under Cases directory
    * a user allows to select a project which they want to delete
    */
        void
projectComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
     if( event.getStateChange() == event.SELECTED ){
        String s = (String)event.getItem();

        projectTextField.setText(casesDirectory+s);
     }
        }

  /**
    * Remove all files under the deleting project before delete the project
folder
    *
    * @param aFile : the deleting project folder
    */
  void removeAll(File aFile){
                String[] l = aFile.list();
                for( int i=0; i<l.length; i++ ){
                   File file = new File(aFile, l[i]);
                if( file.isDirectory() ){
                   removeAll(file);
                }
                else{
                   file.delete();
                }
             }
             aFile.delete();
  }
}
```

```
package Cases;

/**
 * Dependency object which is used to save in the dependency.cfg file
 */

import java.io.Serializable;

////////////////////////////////////////////////////////////////////////
/**
 * Dependency : Create a Dependency object and save it in dependency.cfg
file
 */
////////////////////////////////////////////////////////////////////////
public class Dependency implements Serializable{

  /**
   * loopName : evolution process name
   */
  private String loopName = null;

  /**
```

```
   * step : step type name
   */
  private String step = null;

  /**
   * outputComponent : output component of the step type
   */
  private String outputComponent = null;

  /**
   * primaryInput : primary input of the step type
   */
  private String primaryInput = null;

  /**
   * secondaryInput : secondary input of the step type
   */
  private String secondaryInput = null;;

  /**
   * This Dependency constructor is used to create a Dependency object
   */
  public Dependency( String loopName, String step, String
outputComponent, String primaryInput, String secondaryInput ){
      this.loopName = loopName;
      this.step = step;
      this.outputComponent = outputComponent;
      this.primaryInput = primaryInput;
      this.secondaryInput = secondaryInput;
  }

  public Dependency() {}

  /**
   * @return loopName : evolution process name
   */
  public String getLoopName(){
     return this.loopName;
  }

  /**
   * @return step : step type name
   */
  public String getStep(){
     return this.step;
  }

  /**
   * @return outputComponent : output component of the step type
   */
  public String getOutputComponent(){
     return this.outputComponent;
  }

  /**
   * @return primaryInput : primary input of the step type
   */
  public String getPrimaryInput(){
     return this.primaryInput;
  }

  /**
   * @return secondaryInput : secondary input of the step type
   */
  public String getSecondaryInput(){
     return this.secondaryInput;
  }

  /**
   * Set secondary input for this Dependency object
   *
   * @param secondaryInput : secondary input component
   */
  public void setSecondaryInput( String secondaryInput ){
```

```
    this.secondaryInput = secondaryInput;
  }
}
```

```
package Cases;

import java.awt.*;
import java.io.*;
import java.util.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * EditDecomposeFrame : use to edit/decompose a selected step
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_EditDecompose
 */
////////////////////////////////////////////////////////////////////////
public class EditDecomposeFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_EditDecompose
{
  /**
   * pathName : an absolute path of a selected step
   */
  String pathName = null;

  /**
   * newPath : an absolute path after decompose a selected step
   */
  String newPath = null;

  /**
   * isEdit : a flag to keep track the purpose of using this frame
   */
  boolean isEdit = false;

  /**
   * Build EditDecomposeFrame
   */
      public EditDecomposeFrame()
      {
          // This code is automatically generated by Visual Cafe
when you add
          // components to the visual environment. It instantiates
and initializes
          // the components. To modify the code, only use code
syntax that matches
          // what Visual Cafe can generate, or Visual Cafe may be
unable to back
          // parse your Java file into its visual environment.
          //{{INIT_CONTROLS
          setTitle("Edit");
          getContentPane().setLayout(null);
          setSize(470,305);
          setVisible(false);

      titleLabel.setVerticalTextPosition(com.sun.java.swing.SwingConst
ants.BOTTOM);

      titleLabel.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.RIGHT);

      titleLabel.setVerticalAlignment(com.sun.java.swing.SwingConsta
nts.BOTTOM);
          titleLabel.setText("TESTING: ");
          getContentPane().add(titleLabel);
          titleLabel.setForeground(java.awt.Color.black);
```

```
        titleLabel.setFont(new Font("Dialog", Font.BOLD, 18));
        titleLabel.setBounds(3,6,120,30);

    JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel1.setText("Step Version");
        getContentPane().add(JLabel1);
        JLabel1.setForeground(java.awt.Color.black);
        JLabel1.setBounds(7,54,175,22);
        stepTextField.setEditable(false);
        getContentPane().add(stepTextField);
        stepTextField.setBackground(java.awt.Color.white);
        stepTextField.setBounds(187,54,270,22);

    JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel2.setText("Output Component");
        getContentPane().add(JLabel2);
        JLabel2.setForeground(java.awt.Color.black);
        JLabel2.setBounds(7,96,175,22);
        outputTextField.setEditable(false);
        getContentPane().add(outputTextField);
        outputTextField.setBackground(java.awt.Color.white);
        outputTextField.setBounds(187,96,270,22);

    JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel3.setText("Primary Input Component(s)");
        getContentPane().add(JLabel3);
        JLabel3.setForeground(java.awt.Color.black);
        JLabel3.setBounds(7,138,175,22);
        getContentPane().add(primaryTextField);
        primaryTextField.setBounds(187,138,270,22);

    JLabel4.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel4.setText("Secondary Input Component(s)");
        getContentPane().add(JLabel4);
        JLabel4.setForeground(java.awt.Color.black);
        JLabel4.setBounds(7,180,175,22);
        getContentPane().add(secondaryTextField);
        secondaryTextField.setBounds(187,180,270,22);

    selectedLabel.setVerticalTextPosition(com.sun.java.swing.SwingC
onstants.BOTTOM);

    selectedLabel.setVerticalAlignment(com.sun.java.swing.SwingCo
nstants.BOTTOM);
        selectedLabel.setText("Selected Label");
        getContentPane().add(selectedLabel);
        selectedLabel.setBackground(new
java.awt.Color(204,204,204));
        selectedLabel.setForeground(java.awt.Color.black);
        selectedLabel.setFont(new Font("Dialog", Font.BOLD,
16));
        selectedLabel.setBounds(130,6,290,30);
        saveButton.setText("Save");
        saveButton.setActionCommand("Save");
        getContentPane().add(saveButton);
        saveButton.setBounds(12,258,75,22);
        exitButton.setText("Cancel");
        exitButton.setActionCommand("Save");
        getContentPane().add(exitButton);
        exitButton.setBounds(372,258,75,22);
        getContentPane().add(deleteButton);
        deleteButton.setBounds(0,0,0,0);
        //}}

        //{{INIT_MENUS
        //}}

        //{{REGISTER_LISTENERS
        SymAction lSymAction = new SymAction();
        saveButton.addActionListener(lSymAction);
```

```
                deleteButton.addActionListener(lSymAction);
                exitButton.addActionListener(lSymAction);
                //}}
        }
    /**
     * CasesFrame launch this frame through SPIDER --> Edit
     */
        public EditDecomposeFrame(String sTitle, String stepName,
                        String output, String pathName)
        {
                this();
      //Set Delete button for Edit frame
                deleteButton.setText("Delete");
                deleteButton.setActionCommand("Delete");
                getContentPane().add(deleteButton);
                deleteButton.setBounds(84,258,75,22);

                setTitle(sTitle);
                this.titleLabel.setText( "Edit : " );
                this.selectedLabel.setText( pathName );
                this.stepTextField.setText( stepName );
                this.outputTextField.setText( output );
                this.pathName = pathName;

                try{
                   File f = new File(pathName+"\\input.p");
                   if( f.exists() ){
                      DataInputStream primIn = new DataInputStream(
new FileInputStream(f));
                      if( primIn != null ){
                         this.primaryTextField.setText(
(String)primIn.readLine() );
                      }
        }
        f = new File(pathName+"\\input.s");
        if( f.exists() ){
                      DataInputStream secondIn = new DataInputStream(
new FileInputStream(f));
                      if( secondIn != null ){
                         this.secondaryTextField.setText(
(String)secondIn.readLine() );
                      }
        }
                }
                catch( IOException io ){ System.out.println(io);}

                newPath = this.pathName;
                isEdit = true;
        }

    /**
     * CasesFrame launch this frame through SPIDER --> Decompose
     */
        public EditDecomposeFrame(String sTitle, String stepName,
                        String output, String pathName, int subDir)
        {
                this();

                setTitle(sTitle);
                this.titleLabel.setText( "Decompose:" );
                this.selectedLabel.setText( pathName);
                this.stepTextField.setText( stepName );
                this.outputTextField.setText( output );
                this.pathName = pathName;

                try{
                   File f = new File(pathName+"\\input.p");
                   if( f.exists() ){
                      DataInputStream primIn = new DataInputStream(
new FileInputStream(f));
                      if( primIn != null ){
                         this.primaryTextField.setText(
(String)primIn.readLine() );
```

```
                }
        }
        f = new File(pathName+"\\input.s");
        if( f.exists() ){
                DataInputStream secondIn = new DataInputStream(
new FileInputStream(f));
                if( secondIn != null ){
                   this.secondaryTextField.setText(
(String)secondIn.readLine() );
                }
        }
                }
                catch( IOException io ){ System.out.println(io);}

                newPath = this.pathName + "\\" + subDir;
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new EditDecomposeFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();
                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel titleLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField stepTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField outputTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField primaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel4 = new
com.sun.java.swing.JLabel();
```

```
        com.sun.java.swing.JTextField secondaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel selectedLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton saveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton exitButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton deleteButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == saveButton)
                                saveButton_actionPerformed(event);
                        else if (object == deleteButton)
                                deleteButton_actionPerformed(event);
                        else if (object == exitButton)
                                exitButton_actionPerformed(event);
                }
        }

  /**
   * Save all the creations or the changes after decomposing and editting
   */
        public void
saveButton_actionPerformed(java.awt.event.ActionEvent event)
        {
```



```
                File dir = new File(newPath);
                if( !dir.exists() ){
                  dir.mkdir();
                }

                if( dir.isDirectory() ){
              try{
                if( !isEdit ){
                        File oldFile = new File(this.pathName,
COMPONENT_CONTENT_DIR);
                        File newFile = new
File(newPath,COMPONENT_CONTENT_DIR);
                    newFile.mkdir();
                    if( oldFile.exists() && newFile.isDirectory() ){
                      copyFile(oldFile, newFile);
                    }
                  }
                        if( checkInputs(secondaryTextField.getText())) {
                    FileOutputStream fileOutput = new FileOutputStream(new
File(dir, "\\input.p"));
                    DataOutputStream depPrimary = new DataOutputStream(
fileOutput);
                    if( depPrimary != null ){
                      if( !primaryTextField.getText().equals("") ){
                        depPrimary.writeBytes(primaryTextField.getText());
                      }
                    }
                    depPrimary.flush();
                    depPrimary.close();
                    fileOutput.close();

                    fileOutput = new FileOutputStream(new File(dir,"\\input.s" ));
                    DataOutputStream depSecondary = new DataOutputStream(
fileOutput );
                    if( depSecondary != null ){
                      if( !secondaryTextField.getText().equals("") ){
                        depSecondary.writeBytes(secondaryTextField.getText());
```

```
                }
            }
            depSecondary.flush();
            depSecondary.close();
            fileOutput.close();
            exitButton_actionPerformed(null);
          }
        }
        catch(FileNotFoundException fex ){ System.out.println(fex);}
        catch(IOException e){System.out.println(e);}
      }
          }

  /**
   * Delete this selected step
   */
          void deleteButton_actionPerformed(java.awt.event.ActionEvent
event)
          {
            int result = JOptionPane.showConfirmDialog( this, "All the files
and subdirectories will be deleted! Would you like to continue?",
                          "Confirm
Message",JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
      //result = 0 ==> yes
      //result = 1 ==> no
      if( result == 0 ){
        File aFile = new File(this.pathName);
        if( aFile.exists() ){
          removeAll(aFile);
        }
      }
      exitButton_actionPerformed(null);
          }

  /**
   * Exit EditDecomposeFrame
   */
          public void
exitButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            this.setEnabled(false);
    dispose();
          }

  /**
   * Copy all link files under Component Content directory into the new
decomposed step
   *
   * @param oldFile : old link files of a parent file
   * @param newFile : new link files of new atomic
   */
          public void copyFile( File oldFile, File newFile ){
            try{
              for( int i=0; i< LINK_FILE_NAMES.length; i++ ){
            File oldLink = new File(oldFile, LINK_FILE_NAMES[i]);
            File newLink = new File(newFile, LINK_FILE_NAMES[i]);
            FileWriter fileWriter = new FileWriter (newLink);
            BufferedWriter bw = new BufferedWriter( fileWriter );
            if( bw != null ){
              String line = null;
              FileReader fileReader = new FileReader(oldLink);
              BufferedReader br = new BufferedReader( fileReader);
              if( br != null ){
                while( (line=br.readLine()) != null ){
                  bw.write(line);
                }
              }
              br.close();
              fileReader.close();
            }
            bw.flush();
            bw.close();
            fileWriter.close();
```

```
        }
      }
      catch( FileNotFoundException f){System.out.println(f);}
      catch(IOException io ){System.out.println(io);}
          }

    /**
      * Check all components in secondary textfield are valid or not
      * @param secondary : a string of secondaryTextField
      * @return boolean : valid or invalide input
      */
          public boolean checkInputs(String secondary ){
            String output = outputTextField.getText();
            String sub2 = null;

            if( secondary != null ){
              StringTokenizer st = new StringTokenizer(secondary, ",");
              while( st.hasMoreTokens() ){
                String s = (String)st.nextToken();
           int j = s.indexOf("-");
           if( j>0){
                    sub2 = (s.substring(j+1)).trim();
             if( !sub2.equals(output) ){
                      char c = (char)sub2.charAt(0);
                      boolean isLetter = (new Character(c)).isLetter(c);
                      if( isLetter ){
                        JOptionPane.showMessageDialog(this, s+" is
invalid component!",
                                      "Error
Message",JOptionPane.ERROR_MESSAGE);
                  return false;
                  }
               }
            }
          }
      }
            return true;
        }

    /**
      * Remove all files under the deleting the selected step before delete the
step folder
      * @param aFile : the deleting the selected step folder
      */
    void removeAll(File aFile){
                  String[] l = aFile.list();
                  for( int i=0; i<l.length; i++ ){
                    File file = new File(aFile, l[i]);
          if( file.isDirectory() ){
            removeAll(file);
                  }
                  else{
                    file.delete();
                  }
                }
                aFile.delete();
    }
}
```

```
package Cases;

import java.io.Serializable;

////////////////////////////////////////////////////////////////////////
/**
 * EHL : Create a EHL object and save it in loop.cfg file
 */
////////////////////////////////////////////////////////////////////////
public class EHL implements Serializable{

  /**
   * EHLName : evolution history process name
   */
  private String EHLName;

  /**
   * EHLPath : a string of step types in the evolution process
   */
  private String EHLPath;

  /**
   * This EHL constructor is used to create a EHL object
   */
  public EHL( String EHLName, String EHLPath ){
    if( EHLName == null ){
      EHLName = "";
    }

    if( EHLPath == null ){
      EHLPath = "";
    }
    this.EHLName = EHLName;
    this.EHLPath = EHLPath;
  }

  public EHL() {}

  /**
   * @return EHLName : evolution history process name
   */
  public String getEHLName(){
    return this.EHLName;
  }

  /**
   * @return EHLPath : a string of step types in the evolution process
   */
  public String getEHLPath(){
    return this.EHLPath;
  }
}
```



```
package Cases;

import java.util.*;
import java.io.*;

////////////////////////////////////////////////////////////////////////
/**
 * I_AVC : an interface for 3 classes - AVCCreateStepFrame
 *                                    - AVCMergingFrame
 *                                    - AVCSplittingFrame
 */
////////////////////////////////////////////////////////////////////////
public interface I_AVC {
  public void checkPath(Vector v);
  public void createFiles( Dependency dep, File theFile);
        public void setLoopNameComboBox( Vector loopVector );
        public Vector tokenizeVector(String string);
}
```

```
package Cases;

import java.util.*;
import java.io.*;

////////////////////////////////////////////////////////////////////////
/**
 * I_AVCOpenStep : an interface for AVCOpenStepFrame
 */
////////////////////////////////////////////////////////////////////////
public interface I_AVCOpenStep{
   public void setInitialFrame( VersionControl vc );
   public void setLoopNameComboBox( Vector loopVector );
   public void setStepComboBox( Vector stepVector );
   public void setVersionComboBox( Vector versionVector );

}
```



```
package Cases;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

////////////////////////////////////////////////////////////////////////
/**
 * I_Cases : an interface for CasesFrame
 */
////////////////////////////////////////////////////////////////////////
public interface I_Cases{
        public void getVersionControl();
        public void directoryTree();
   public void currentDir(String absPath );
   public void subDir(String absPath );
   public Vector getComponents(String stepName, String absPath);
        public void setComponentContent( String stepName, String
absPath );
   public void setStepContent(TraceFrame traceFrame, String stepName,
String absPath );
   public void setTraceFrame(String stepName, String absPath);
   public void getAtomics(File aFile, Vector storedVector);
        public int findMax( String thePath );
        public void tokenizer( Vector v, String s);
        public Vector fileNameList();
        public void savePersonnelVector(Vector v);
        public Vector getPersonnelVector();
        public Vector getAllAtomics();
        public void saveStepContentVector(Vector v);
        public Vector getStepContentVector();
        public Vector getScheduledAtomicVector();
}
```

```
package Cases;

import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * I_ComponentContent : an interface for ComponentContentFrame
 */
////////////////////////////////////////////////////////////////////////
public interface I_ComponentContent{
   public String checkInput(String selectedItem );
   public JComboBox findComboBox( String fileType );
   public void saveLinkFile( String fileType);
```

```
        public void searchFiles(File aFile, String fileType);
    public File searchPath( String s);
    public void setTextLink(Vector v);
    public void setWordLink(Vector v);
    public void setExcelLink(Vector v);
    public void setDataLink(Vector v);
    public void setURLLink(Vector v);
    public void setCAPSLink(Vector v);
}
```

```
package Cases;

import java.io.*;

////////////////////////////////////////////////////////////////////////
/**
  * I_EditDecompose : an interface for EditDecomposeFrame
  */
////////////////////////////////////////////////////////////////////////
public interface I_EditDecompose{
        public boolean checkInputs(String secondary );
        public void copyFile( File oldFile, File newFile );
}
```



```
package Cases;

import java.util.*;

////////////////////////////////////////////////////////////////////////
/**
  * I_Personnel : an interface for PersonnelFrame
  */
////////////////////////////////////////////////////////////////////////
public interface I_Personnel{
        public void setInitial(Personnel personnel);
        public Personnel getPersonnelData();
    public void setSkillComboBox(Vector v);
}
```

```
package Cases;

import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import com.sun.java.swing.event.*;

////////////////////////////////////////////////////////////////////////
/**
  * I_ProjectSchema : an interface for ProjectSchemaFrame
  */
////////////////////////////////////////////////////////////////////////
public interface I_ProjectSchema{
    public void readDepFile();
    public void readInputFiles( String pathName );
    public void setCompComboBox( Vector compVector );
```

```
    public void setCompInfo( ComponentType ct );
    public void setDepenEHLComboBox( Vector EHLVector );
    public void setDependStepComboBox( Vector stepVector );
    public void setDepInfo( String selectedDepStep);
    public void setEHLComboBox( Vector EHLVector );
    public void setEHLInfo( EHL ehl );
    public void setListModel( Vector stepVector );
    public void setStepComboBox( Vector stepVector );
        public void setItemList(Object[] objs);
}
```



```
package Cases;

import java.util.*;

////////////////////////////////////////////////////////////////////
/**
 * I_StepContent : an interface for StepContentFrame
 */
////////////////////////////////////////////////////////////////////
public interface I_StepContent{
        public StepContent getStepContent();
        public void setInitial( StepContent stepContent );
    public void setPredecessorComboBox(Object[] oa);
        public void setReadOnly();
        public void setStakeHolders();
    public void setSkillComboBox(Vector v);
}
```

```
package Cases;

import java.util.*;
import java.io.*;

////////////////////////////////////////////////////////////////////
/**
 * I_Trace : an interface for TraceFrame
 */
////////////////////////////////////////////////////////////////////
public interface I_Trace{
    public String checkSelection(String selectedItem );
    public String convertToThePath(String s);
    public File searchFilePath( String s);
        public void searchFiles(File aFile, String fileType);
    public void searchIndex();
    public void searchInputFiles( String thePath );
    public void searchPath( String s);
        public void setComponentContent(String selectedItem, File f);
    public void setHistory( String s);
    public void setInitial(Vector componentsVector);
    public void setSelectedItem(String currentPath, String selectedItem);
        public void tokenizer( Vector v, String s);
}
```

```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * ListDialog : Use to list components.  It is launched by
StepContentFrame,
 *          TraceFrame, and ProjectSchemaFrame.
 *
 * Implement CasesTitle where stores all global variables of Cases package
 */
////////////////////////////////////////////////////////////////////////
public class ListDialog extends com.sun.java.swing.JDialog implements
CasesTitle
{
  /**
   * tf : trace frame, one of owner frames
   */
  TraceFrame tf = null;

  /**
   * scf : step component frame, one of owner frames
   */
  StepContentFrame scf = null;

  /**
   * psf : project schema frame, one of owner frames
   */
  ProjectSchemaFrame psf = null;

  /**
   * listModel : model of the itemList
   */
  DefaultListModel listModel = new DefaultListModel();

  /**
   * index : index of component content or trace button from TraceFrame
   */
  int index = 0;

  /**
   * Buil ListDialog
   */
        public ListDialog(JFrame parent)
        {
                super(parent);

                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setModal(true);
                getContentPane().setLayout(null);
                setSize(300,400);
                setVisible(false);
                getContentPane().add(itemScrollPane);
                itemScrollPane.setBounds(15,70,270,250);
                itemScrollPane.getViewport().add(itemList);
                itemList.setBounds(0,0,267,247);
                OKButton.setText("OK");
                getContentPane().add(OKButton);
                OKButton.setBounds(64,340,75,24);
                cancelButton.setText("Cancel");
                getContentPane().add(cancelButton);
```

```
                cancelButton.setBounds(160,340,75,24);

        titleLabel.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.CENTER);
                titleLabel.setText("jlabel");
                getContentPane().add(titleLabel);
                titleLabel.setForeground(java.awt.Color.black);
                titleLabel.setFont(new Font("Dialog", Font.BOLD, 18));
                titleLabel.setBounds(1,10,298,30);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                //}}

                /**
                 * Set model for the itemList
                 */
                itemList.setModel(listModel);
        }

  /**
   * StepContentFrame launchs this dialog with its components list
   *
   * @param scf : step content frame, owner frame
   * @param title : title of this dialog when step content frame launchs it
   * @param itemVector : list of components from step content frame
   */
        public ListDialog(StepContentFrame scf, String title, Vector
itemVector)
        {
                this((JFrame)scf);
                this.scf = scf;
                setTitle(title);
                this.titleLabel.setText(title);
                for( int i=0; i<itemVector.size(); i++ ){
                  listModel.addElement(itemVector.elementAt(i));
                }
        }

  /**
   * TraceFrame launchs this dialog with its components list
   *
   * @param tf : trace frame, owner frame
   * @param title : title of this dialog when trace frame launchs it
   * @param itemVector : list of components from trace frame
   * @param index : index of button from trace frame, eg, 0:trace button,
or
   *                                  1:component content buton
   */
        public ListDialog(TraceFrame tf, String title, Vector itemVector,
int index)
        {
                this((JFrame)tf);
                this.tf = tf;
                this.index = index;
                setTitle(title);
                this.titleLabel.setText(title);
                for( int i=0; i<itemVector.size(); i++ ){
                  listModel.addElement(itemVector.elementAt(i));
                }
        }

  /**
   * ProjectSchemaFrame launchs this dialog with its components list
   *
   * @param psf : project schema frame, owner frame
   * @param title : title of this dialog when project schema frame launchs
it
   * @param itemVector : list of components from project schema frame
   */
```

```
        public ListDialog(ProjectSchemaFrame psf, String title, Vector
itemVector)
        {
                this((JFrame)psf);
                this.psf = psf;
                setTitle(title);
                this.titleLabel.setText(title);
                for( int i=0; i<itemVector.size(); i++ ){

listModel.addElement(((ComponentType)itemVector.elementAt(i)).getCom
ponentID());
                }
        }

        public void setVisible(boolean b)
        {
                if (b){
                   this.setLocationRelativeTo(this.scf);
                }
                super.setVisible(b);
        }


        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets
                Insets insets = getInsets();
                        setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JScrollPane itemScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JList itemList = new
com.sun.java.swing.JList();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel titleLabel = new
com.sun.java.swing.JLabel();
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                }
        }

  /**
   * Return selected item(s) to the owner frame
   */
```

```
        void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    if( this.scf != null ){

this.scf.setPredecessorComboBox(this.itemList.getSelectedValues());
      setVisible(false);
      dispose();
    }
    else if( this.tf != null ){
      String selectedItem = (String)this.itemList.getSelectedValue();
      if( index == 0 ){
        String s = this.tf.convertToThePath(selectedItem);
        if( s!= null ){
          this.tf.setSelectedItem(s, selectedItem);
          setVisible(false);
          dispose();
        }
      }
      else if( index == 1 ){
        String currentComponent =
(String)this.tf.checkSelection(selectedItem);
        if( !currentComponent.equals("") ){
          File f = (File)this.tf.searchFilePath(currentComponent);
          if( !f.exists() ){
            JOptionPane.showMessageDialog(this, selectedItem+" does
not exist!", "Alert Message",
                                JOptionPane.ERROR_MESSAGE);
          }
          else{
            String s = this.tf.convertToThePath(currentComponent);
            if( s!= null ){
              this.tf.setSelectedItem(s, selectedItem);
              setVisible(false);
              dispose();
              this.tf.setComponentContent(selectedItem, f);
            }
            setVisible(false);
            dispose();
          }
        }
      }
    }
    else if( this.psf != null ){
      this.psf.setItemList(this.itemList.getSelectedValues());
      setVisible(false);
      dispose();
    }
      }

  /**
   * Exit ListDialog
   */
      void cancelButton_actionPerformed(java.awt.event.ActionEvent
event)
      {
    setVisible(false);
    dispose();
      }
}
```

```
package Cases;

/**
 * Personnel Data object which is used to save
 * under stakeholder directory
 */

import java.io.Serializable;
import java.util.*;

////////////////////////////////////////////////////////////////////////
/**
 * Personnel : Create a Personnel object and save it in a file with
 *          the name is Personnel's ID under stakeholder directory
 */
////////////////////////////////////////////////////////////////////////
public class Personnel implements Serializable{

   /**
    * ID : personnel ID
    */
   private String ID = null;

   /**
    * name : name of a person
    */
   private String name = null;

   /**
    * skill : vector of the person's skills
    */
   private Vector skill = new Vector();

   /**
    * securityLevel : security level of the person
    */
   private int  securityLevel = 0;

   /**
    * email : e-mail address of the person
    */
   private String  email = null;

   /**
    * telephone : telephone number of the person
    */
   private String  telephone = null;

   /**
    * fax : fax number of the person
    */
   private String  fax = null;

   /**
    * address : address of the person
    */
   private String address = null;

   /**
    * majorJobs : major job list for this person
    */
   private Vector majorJobs = new Vector();

   /**
    * minorJobs : minor job list for this person
    */
   private Vector minorJobs = new Vector();

   /**
    * This Personnel constructor is used to create a Personnel object
    */
   public Personnel(String ID, String name, Vector skill,
                    int securityLevel, String email, String telephone,
                    String fax, String address){
```

```
    this();
    this.ID = ID;
    this.name = name;
    this.skill = skill;
    this.securityLevel = securityLevel;
    this.email = email;
    this.telephone = telephone;
    this.fax = fax;
    this.address = address;
}

/**
 * Empty Personnel constructor
 */
public Personnel(){
}

/**
 * Set ID for this personnel object
 *
 * @param s : string of ID
 */
public void setID(String s){
    this.ID = s;
}

/**
 * ID of this personnel object
 *
 * @return ID : this person's ID
 */
public String getID(){
    return this.ID;
}

/**
 * Set name for this personnel object
 *
 * @param s : this person's name
 */
public void setName(String s){
    this.name = s;
}

/**
 * The person's name
 *
 * @return name : the name of person
 */
public String getName(){
    return this.name;
}

/**
 * Set skills for the person
 *
 * @param v : list of person's skill
 */
public void setSkill(Vector v){
    this.skill = v;
}

/**
 * Skills of the person
 *
 * @return skill : a vector of skills
 */
public Vector getSkill(){
    return this.skill;
}

/**
 * Set security level for the person
 *
```

```
 * @param i : security level
 */
public void setSecurityLevel(int i){
   this.securityLevel = i;
}

/**
 * Security level of the person
 *
 * @return securityLevel : security level
 */
public int getSecurityLevel(){
   return this.securityLevel;
}

/**
 * Set email for the person
 *
 * @param s : email address of the person
 */
public void setEmail(String s){
   this.email = s;
}

/**
 * Email address of the person
 *
 * @return email : email address of the person
 */
public String getEmail(){
   return this.email;
}

/**
 * Set telephone number of the person
 *
 * @param s : a string of telephone number
 */
public void setTelephone(String s){
   this.telephone = s;
}

/**
 * Telephone number of the person
 *
 * @return telephone : a string of telephone number
 */
public String getTelephone(){
   return this.telephone;
}

/**
 * Set fax number for the person
 *
 * @param s : a string of fax number
 */
public void setFax(String s){
   this.fax = s;
}
/**
 * Fax number of the person
 *
 * @return s : a string of fax number
 */
public String getFax(){
   return this.fax;
}

/**
 * Set address for the person
 *
 * @param s : a string of address
 */
public void setAddress(String s){
```

```
    this.address = s;
}

/**
 * Address of the person
 *
 * @return address : a string of address
 */
public String getAddress(){
    return this.address;
}
/**
 * Set the list of minor jobs for the person
 *
 * @param v : a list of minor jobs
 */
public void setMinorJobs(Vector v){
    this.minorJobs = v;
    for( int i=0; i<v.size(); i++ ){
        System.out.println(i+" : MINORJOB = "+v.elementAt(i));
    }
}
/**
 * The list of minor jobs for the person
 *
 * @return minorJobs : a vector of minor jobs
 */
public Vector getMinorJobs(){
    return this.minorJobs;
}
/**
 * Set the list of major jobs for the person
 *
 * @param v: a vector of major jobs
 */
public void setMajorJobs(Vector v){
    this.majorJobs = v;
    for( int i=0; i<v.size(); i++ ){
        System.out.println(i+" : MAJORJOB = "+v.elementAt(i));
    }
}
/**
 * The list of major jobs for the person
 *
 * @return majorJobs : a vector of major jobs
 */
public Vector getMajorJobs(){
    return this.majorJobs;
}
}



package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;
import symantec.itools.awt.MultiList;
import com.symantec.itools.swing.borders.EtchedBorder;


////////////////////////////////////////////////////////////////////////
/**
 * PersonnelFrame : to create, edit, and view personnel object
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_Personnel
 */
////////////////////////////////////////////////////////////////////////
public class PersonnelFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_Personnel
{
```

```
/**
 * selectedSkill : contains all selected skill from SkillTableFrame
 */
public Vector selectedSkill = new Vector();

/**
 * view : a flag to define the purpose of using this frame, eg. view or edit
 */
boolean view = false;

/**
 * edit : a flag to define the purpose of using this frame, eg. view or edit
 */
boolean edit = false;

Personnel personnel = null;

/**
 * listHeadings : the 3 column headings of job multi list
 */
String[] listHeadings = {"Job Name", "Real Start Time", "Estimated
Duration"};

/**
 * Build PersonnelFrame and CasesFrrame launch this to create
personnel object
 */
    public PersonnelFrame()
    {
        // This code is automatically generated by Visual Cafe
when you add
        // components to the visual environment. It instantiates
and initializes
        // the components. To modify the code, only use code
syntax that matches
        // what Visual Cafe can generate, or Visual Cafe may be
unable to back
        // parse your Java file into its visual environment.
        //{{INIT_CONTROLS
        setTitle("Personnel Data");
        getContentPane().setLayout(null);
    setVisible(false);
    setSize(500,540);

    JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel2.setText("ID");
        getContentPane().add(JLabel2);
        JLabel2.setForeground(java.awt.Color.black);
        JLabel2.setBounds(45,40,110,24);

    JLabel5.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel5.setText("E-mail Address");
        getContentPane().add(JLabel5);
        JLabel5.setForeground(java.awt.Color.black);
        JLabel5.setBounds(45,180,110,24);

    JLabel6.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel6.setText("Fax Number");
        getContentPane().add(JLabel6);
        JLabel6.setForeground(java.awt.Color.black);
        JLabel6.setBounds(45,250,110,24);

    JLabel7.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
        JLabel7.setText("Address");
        getContentPane().add(JLabel7);
        JLabel7.setForeground(java.awt.Color.black);
        JLabel7.setBounds(45,285,110,24);

    JLabel8.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
```

```
                JLabel8.setText("Name");
                getContentPane().add(JLabel8);
                JLabel8.setForeground(java.awt.Color.black);
                JLabel8.setBounds(45,75,110,24);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Security Level");
                getContentPane().add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(45,145,110,24);

        JLabel10.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel10.setText("Telephone Number");
                getContentPane().add(JLabel10);
                JLabel10.setForeground(java.awt.Color.black);
                JLabel10.setBounds(45,215,110,24);

        JLabel11.setHorizontalTextPosition(com.sun.java.swing.SwingCo
nstants.CENTER);

        JLabel11.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.CENTER);
                JLabel11.setText("Personnel Data");
                getContentPane().add(JLabel11);
                JLabel11.setForeground(java.awt.Color.black);
                JLabel11.setFont(new Font("Dialog", Font.BOLD, 20));
                JLabel11.setBounds(0,2,500,40);
                getContentPane().add(IDTextField);
                IDTextField.setBackground(java.awt.Color.white);
                IDTextField.setForeground(java.awt.Color.black);
                IDTextField.setBounds(155,40,300,24);
                getContentPane().add(nameTextField);
                nameTextField.setBackground(java.awt.Color.white);
                nameTextField.setForeground(java.awt.Color.black);
                nameTextField.setBounds(155,75,300,24);
```



```
getContentPane().add(clearButton);
clearButton.setBounds(0,0,0,0);
skillButton.setText("Skill");
skillButton.setActionCommand("Skill");
getContentPane().add(skillButton);
skillButton.setBounds(55,110,100,24);
exitButton.setText("Exit");
exitButton.setActionCommand("Exit");
getContentPane().add(exitButton);
exitButton.setBounds(0,0,0,0);
getContentPane().add(saveButton);
saveButton.setBounds(0,0,0,0);
getContentPane().add(skillComboBox);
skillComboBox.setBounds(155,110,300,24);
getContentPane().add(securityComboBox);
securityComboBox.setBounds(155,145,300,24);
getContentPane().add(emailTextField);
emailTextField.setBackground(java.awt.Color.white);
emailTextField.setForeground(java.awt.Color.black);
emailTextField.setBounds(155,180,300,24);
getContentPane().add(telephoneTextField);
telephoneTextField.setBackground(java.awt.Color.white);
telephoneTextField.setForeground(java.awt.Color.black);
telephoneTextField.setBounds(155,215,300,24);
getContentPane().add(addressTextField);
addressTextField.setBackground(java.awt.Color.white);
addressTextField.setForeground(java.awt.Color.black);
addressTextField.setBounds(155,285,300,24);
getContentPane().add(faxTextField);
faxTextField.setBackground(java.awt.Color.white);
faxTextField.setForeground(java.awt.Color.black);
faxTextField.setBounds(155,250,300,24);
getContentPane().add(JLabel1);
JLabel1.setBounds(0,0,0,0);
getContentPane().add(jobsScrollPane);
jobsScrollPane.setBounds(0,0,0,0);
getContentPane().add(JLabel3);
```

```
                JLabel3.setBounds(0,0,0,0);
                getContentPane().add(jobScrollPane);
                jobScrollPane.setBounds(0,0,0,0);
                onHandsPanel.setLayout(new
FlowLayout(FlowLayout.CENTER,5,5));
                getContentPane().add(onHandsPanel);
                onHandsPanel.setBounds(0,0,0,0);
                getContentPane().add(minorRadioButton);
                minorRadioButton.setBounds(0,0,0,0);
                getContentPane().add(majorRadioButton);
                majorRadioButton.setBounds(0,0,0,0);
                //$$ etchedBorder1.move(0,541);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                skillButton.addActionListener(lSymAction);
                clearButton.addActionListener(lSymAction);
                saveButton.addActionListener(lSymAction);
                exitButton.addActionListener(lSymAction);

                SymItem lSymItem = new SymItem();
                minorRadioButton.addItemListener(lSymItem);
                majorRadioButton.addItemListener(lSymItem);
                //}}

        //Set security level combobox from 0..5
        for( int i=0; i<SECURITY_LEVEL; i++ ){
           securityComboBox.addItem(i+"");
        }
            setSize(500,380);
                clearButton.setText("Clear");
                clearButton.setActionCommand("Delete");
                getContentPane().add(clearButton);
                clearButton.setBounds(24,330,73,24);
                saveButton.setText("Save");
                saveButton.setActionCommand("Save");
                getContentPane().add(saveButton);
                saveButton.setBounds(96,330,73,24);
                exitButton.setBounds(380,330,73,24);
        }

        /**
         * CasesFrame launch this to edit and view personnel object
         *
         * @param selectedFile : a selected file name
         * @param request : purpose of using this frame, eg. View or Edit
         */
        public PersonnelFrame(String selectedFile, String request){
           this();
                        try{
                           File aFile = new File(selectedFile);
                           if( aFile.exists() ){
                              FileInputStream fileInput = new
FileInputStream(aFile);
                              ObjectInputStream oi = new ObjectInputStream(
fileInput );
                              if( oi != null ){
                                 personnel = (Personnel)oi.readObject();
                              }
                              oi.close();
                              fileInput.close();
                }
                        }
                        catch( IOException io ){
                           System.out.println("IOException: "+io);
                        }
                        catch( ClassNotFoundException c ){
                System.out.println("ClassNotFoundException: "+c);
                        }
                        if( request.equals("View") ){
```

```
                view = true;
              setSize(500,540);
                clearButton.setBounds(0,0,0,0);
                saveButton.setBounds(0,0,0,0);

                onHandsPanel.setBorder(etchedBorder1);
                onHandsPanel.setLayout(null);
                getContentPane().add(onHandsPanel);
                onHandsPanel.setBounds(155,320,300,30);
                minorRadioButton.setText("Minor Jobs");
                minorRadioButton.setSelected(true);
                onHandsPanel.add(minorRadioButton);
                minorRadioButton.setBounds(170,3,130,24);
                majorRadioButton.setText("Major Jobs");
                onHandsPanel.add(majorRadioButton);
                majorRadioButton.setBounds(20,3,130,24);
                majorRadioButton.setSelected(true);


JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConstants.RIG
HT);
                JLabel3.setText("On-hand Jobs");
                getContentPane().add(JLabel3);
                JLabel3.setForeground(java.awt.Color.black);
                JLabel3.setBounds(45,320,110,27);
                getContentPane().add(jobScrollPane);
                jobScrollPane.setBounds(45,355,410,130);
                jobScrollPane.getViewport().add(jobMultiList);
                jobMultiList.setBackground(java.awt.Color.white);
                jobMultiList.setBounds(0,0,407,127);
                exitButton.setBounds(384,495,73,24);

                try{
                  jobMultiList.setNumberOfCols(3);
          jobMultiList.setHeadings(listHeadings);
        }
        catch(Exception e){System.out.println(e);}
                }
                else if( request.equals("Edit") ){
                  edit = true;
                }
                if( personnel != null ){
                  setInitial(personnel);
                }
        }

        public PersonnelFrame(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new PersonnelFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
```

```
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        //symantec.itools.awt.MultiList jobMultiList = new
symantec.itools.awt.MultiList();
        MultiList jobMultiList = new MultiList();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel5 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel6 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel7 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel8 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel10 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel11 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField IDTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField nameTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JButton clearButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton skillButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton exitButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton saveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JComboBox skillComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox securityComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextField emailTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField telephoneTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField addressTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField faxTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane jobsScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane jobScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JPanel onHandsPanel = new
com.sun.java.swing.JPanel();
```

```
        com.sun.java.swing.JRadioButton minorRadioButton = new
com.sun.java.swing.JRadioButton();
        com.sun.java.swing.JRadioButton majorRadioButton = new
com.sun.java.swing.JRadioButton();
        com.symantec.itools.swing.borders.EtchedBorder etchedBorder1 =
new com.symantec.itools.swing.borders.EtchedBorder();
        //}}

        //{{DECLARE_MENUS
        //}}

  /**
   * Add all selected skills from SkillTableFrame to skillComboBox
   *
   * @param v : a selected skill vector
   */
  public void setSkillComboBox(Vector v){
    this.selectedSkill = new Vector();
    this.skillComboBox.removeAllItems();
    if( v.size() > 0 ){
          this.skillComboBox.addItem("ID : Name : Level");
            for( int i=0; i<v.size(); i++ ){
              this.skillComboBox.addItem(v.elementAt(i));
         this.selectedSkill.addElement(v.elementAt(i));
            }
    }
        }

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == skillButton)
                                skillButton_actionPerformed(event);
                        else if (object == clearButton)
                                clearButton_actionPerformed(event);
                        else if (object == saveButton)
                                saveButton_actionPerformed(event);
                        else if (object == exitButton)
                                exitButton_actionPerformed(event);
                }
        }

  /**
   * Launch SkillTableFrame
   */
        public void
skillButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          (new SkillTableFrame(this)).setVisible(true);
  }

  /**
   * Get a personnel object and save it under stakeholder
   */
        void saveButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
    Personnel personnel = (Personnel)getPersonnelData();
    if( personnel != null ){
      try{
        FileOutputStream fileOutput = new
FileOutputStream(STAKEHOLDER+"\\"+IDTextField.getText());
        ObjectOutputStream oo = new ObjectOutputStream(fileOutput);
        if( oo != null ){
          oo.writeObject(personnel);
        }
        oo.flush();
        oo.close();
        fileOutput.close();
      }
      catch(IOException io){
```

```
            JOptionPane.showMessageDialog(this, "Sorry, saving is
unsuccessful", "Error Message", JOptionPane.ERROR_MESSAGE);
        }
        setVisible( false );
        dispose();
    }
        }

  /**
   * Refresh all the textfields and combo boxes
   */
        void clearButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      //clear all fields
      IDTextField.setText("");
      nameTextField.setText("");
      skillComboBox.removeAllItems();
      securityComboBox.setSelectedIndex(0);
      emailTextField.setText("");
      telephoneTextField.setText("");
      addressTextField.setText("");
      faxTextField.setText("");
        }

  /**
   * Exit PersonnelFrame
   */
        void exitButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
     setVisible( false);
     dispose();
        }

  /**
   * Create and return a personnel object
   *
   * @return personnel object
   */
        public Personnel getPersonnelData(){
    Personnel personnel = new Personnel();
          try{
            String ID = IDTextField.getText();
            String name = nameTextField.getText();
            int security =
Integer.parseInt((String)securityComboBox.getSelectedItem());
            String email = emailTextField.getText();
            String tel = telephoneTextField.getText();
            String fax = faxTextField.getText();
            String address = addressTextField.getText();

            if( ID.equals("") ||
              name.equals("") ||
              selectedSkill.size() == 0 ||
              email.equals("") ||
              tel.equals("") ||
              fax.equals("") ||
              address.equals("") ){
          JOptionPane.showMessageDialog(this, "You did not complete all
the fields", "Error Message", JOptionPane.ERROR_MESSAGE);
          return null;
            }
            else{
              personnel.setID(ID);
              personnel.setName(name);
              personnel.setSkill(selectedSkill);
              personnel.setSecurityLevel(security);
              personnel.setEmail(email);
              personnel.setTelephone(tel);
              personnel.setAddress(address);
              personnel.setFax(fax);
            }
    }
```

```
    catch(Exception e){}
        return personnel;
        }

  /**
   * Set initial frame when it is launched to view and edit
   *
   * @param personnel : a personnel object
   */
        public void setInitial(Personnel personnel){
          if( personnel != null ){
      IDTextField.setText(personnel.getID());
      nameTextField.setText(personnel.getName());
      setSkillComboBox(personnel.getSkill());

securityComboBox.setSelectedItem(""+personnel.getSecurityLevel());
      emailTextField.setText(personnel.getEmail());
      telephoneTextField.setText(personnel.getTelephone());
      addressTextField.setText(personnel.getAddress());
      faxTextField.setText(personnel.getFax());
          }
          if( view ){
      IDTextField.setEditable(false);
      nameTextField.setEditable(false);
      skillButton.setEnabled(false);
      securityComboBox.setEnabled(false);
      emailTextField.setEditable(false);
      telephoneTextField.setEditable(false);
      addressTextField.setEditable(false);
      faxTextField.setEditable(false);
          }
        }

  class SymItem implements java.awt.event.ItemListener
        {
          public void itemStateChanged(java.awt.event.ItemEvent event)
                {
                        Object object = event.getSource();
                        if (object == minorRadioButton)

        minorRadioButton_itemStateChanged(event);
                        else if (object == majorRadioButton)

        majorRadioButton_itemStateChanged(event);
                }
        }

  /**
   * View minor jobs of selected personnel object to the list
   */
  public void
minorRadioButton_itemStateChanged(java.awt.event.ItemEvent event)
        {
    if( event.getStateChange() == event.SELECTED ){
      majorRadioButton.setSelected(false);
      jobMultiList.clear();
      Vector v = new Vector();
      if( personnel != null ){
        v = (Vector)personnel.getMinorJobs();
        for( int j=0; j<v.size(); j++ ){
          StepContent sc = (StepContent)v.elementAt(j);
          System.out.println("stepNamme = "+sc.getStepName());
          jobMultiList.addTextCell(j, 0, sc.getStepName());
          jobMultiList.addTextCell(j, 1, sc.getRealStartTime());
          jobMultiList.addTextCell(j, 2, ""+sc.getDuration());
        }
      }
    }
        }

  /**
   * View major jobs of selected personnel object to the list
   */
```

```
        public void
majorRadioButton_itemStateChanged(java.awt.event.ItemEvent event)
        {
    if( event.getStateChange() == event.SELECTED ){
      minorRadioButton.setSelected(false);
      jobMultiList.clear();
      Vector v = new Vector();
      if( personnel != null ){
        v = (Vector)personnel.getMajorJobs();
        for( int j=0; j<v.size(); j++ ){
          StepContent sc = (StepContent)v.elementAt(j);
          System.out.println("stepNamme = "+sc.getStepName());
          jobMultiList.addTextCell(j, 0, sc.getStepName());
          jobMultiList.addTextCell(j, 1, sc.getRealStartTime());
          jobMultiList.addTextCell(j, 2, ""+sc.getDuration());
        }
      }
    }
        }
}
```



```
package Cases;

import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import com.sun.java.swing.event.*;

////////////////////////////////////////////////////////////////////
/**
 * ProjectSchemaFrame : allows a user to create step types, component
types,
 * evolution history loop, and dependency which are stored in step.cfg,
 * component.cfg, loop.cfg, and dependency.cfg respectively
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_ProjectSchema
 */
////////////////////////////////////////////////////////////////////
public class ProjectSchemaFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_ProjectSchema
{
  /**
   * saveTab : flags of 4 tabs and to confirm that every tab should be saved
before exit the frame
   */
  boolean[] saveTab = {true, true, true, true};

  /**
   * stepVector : contains all step type objects
   */
  public Vector stepVector = new Vector();

  /**
   * compVector : contains all component type objects
   */
```

```
public Vector compVector = new Vector();

/**
 * EHLVector : contains all EHL objects
 */
public Vector EHLVector = new Vector();

/**
 * stepHashtable : with the keys and the objects are step type IDs and
step type objects
 */
public Hashtable stepHashtable = new Hashtable();

/**
 * compHashtable : with the keys and the objects are component type
IDs and component type objects
 */
public Hashtable compHashtable = new Hashtable();

/**
 * EHLHashtable : with the keys and the objects are EHL IDs and EHL
objects
 */
public Hashtable EHLHashtable = new Hashtable();

/**
 * depenHashtable : with the keys and the objects are dependency IDs
and dependency objects
 */
public Hashtable depenHashtable = new Hashtable();

/**
 * testIndex :  to keep track the secondary input of each dependency
object to be set correctly
 */
public int testIndex = 0;

/**
 * selectedStepIndex : to make sure that getting the correct step type
object
 */
public int selectedStepIndex;

/**
 * selectedCompIndex : to make sure that getting the correct component
type object
 */
public int selectedCompIndex;

/**
 * selectedEHLIndex : to make sure that getting the correct EHL object
 */
public int selectedEHLIndex;

/**
 * selectedDepenStepIndex : to make sure that getting the correct
dependency object
 */
public int selectedDepenStepIndex;

/**
 * pathName : the path of the current project
 */
public String pathName;

/**
 * listModel : monitor all step types in EHL panel
 */
public DefaultListModel listModel = new DefaultListModel();

/**
 * Build ProjectSchemaFrame
 */
        public ProjectSchemaFrame()
```

```
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Project Schema");
                getContentPane().setLayout(null);
                setSize(600,450);
                setVisible(false);
                mainPanel.setLayout(new GridLayout(1,1,0,0));
                getContentPane().add(mainPanel);
                mainPanel.setBounds(10,0,580,360);
                mainPanel.add(configManagTabbedPane);

        configManagTabbedPane.setForeground(java.awt.Color.black);
                configManagTabbedPane.setBounds(0,0,580,360);
                stepTypePanel.setLayout(null);
                configManagTabbedPane.add(stepTypePanel);
                stepTypePanel.setBounds(2,27,575,330);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel1.setText("Step Type ID");
                stepTypePanel.add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(62,70,130,22);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel2.setText("Step Type Name");
                stepTypePanel.add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2:setBounds(62,110,130,22);

        JLabel3:setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel3.setText("Step Type Description");
                stepTypePanel.add(JLabel3);
                JLabel3.setForeground(java.awt.Color.black);
                JLabel3.setBounds(62,190,130,22);

        JLabel4.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel4.setText("Existing Step Types");
                stepTypePanel.add(JLabel4);
                JLabel4.setForeground(java.awt.Color.black);
                JLabel4.setBounds(62,150,130,22);
                stepAddButton.setText("Add");
                stepAddButton.setActionCommand("jbutton");
                stepTypePanel.add(stepAddButton);
                stepAddButton.setBounds(12,297,75,22);
                stepDeleteButton.setText("Delete");
                stepDeleteButton.setActionCommand("jbutton");
                stepTypePanel.add(stepDeleteButton);
                stepDeleteButton.setBounds(168,297,75,22);
                stepClearButton.setText("Clear");
                stepClearButton.setActionCommand("jbutton");
                stepTypePanel.add(stepClearButton);
                stepClearButton.setBounds(246,297,75,22);
                stepEditButton.setText("Edit");
                stepEditButton.setActionCommand("jbutton");
                stepTypePanel.add(stepEditButton);
                stepEditButton.setBounds(90,297,75,22);
                stepTypePanel.add(stepIDTextField);
                stepIDTextField.setBounds(195,70,300,22);
                stepTypePanel.add(stepNameTextField);
                stepNameTextField.setBounds(195,110,300,22);
                stepTypePanel.add(existedStepComboBox);
```

```
                existedStepComboBox.setBounds(195,150,300,22);
                stepDescriptionTextArea.setLineWrap(true);
                stepDescriptionTextArea.setWrapStyleWord(true);
                stepTypePanel.add(stepDescriptionTextArea);
                stepDescriptionTextArea.setBounds(195,190,300,80);
                stepSaveButton.setText("Save");
                stepSaveButton.setActionCommand("Save");
                stepTypePanel.add(stepSaveButton);
                stepSaveButton.setBounds(492,297,75,22);

        JLabel16.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel16.setText("Project Label:");
                stepTypePanel.add(JLabel16);
                JLabel16.setForeground(java.awt.Color.black);
                JLabel16.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel16.setBounds(2,6,280,24);
                stepLabel.setText("Label");
                stepTypePanel.add(stepLabel);
                stepLabel.setForeground(java.awt.Color.black);
                stepLabel.setFont(new Font("Dialog", Font.BOLD, 18));
                stepLabel.setBounds(292,6,280,24);
                componentTypePanel.setLayout(null);
                configManagTabbedPane.add(componentTypePanel);
                componentTypePanel.setBackground(new
java.awt.Color(204,204,204));
                componentTypePanel.setBounds(2,27,575,330);
                componentTypePanel.setVisible(false);

        JLabel5.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel5.setText("Component Type ID");
                componentTypePanel.add(JLabel5);
                JLabel5.setForeground(java.awt.Color.black);
                JLabel5.setBounds(75,70,145,22);

        JLabel6.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel6.setText("Component Type Name");
                componentTypePanel.add(JLabel6);
                JLabel6.setForeground(java.awt.Color.black);
                JLabel6.setBounds(75,110,144,22);

        JLabel7.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel7.setText("Component Type Description");
                componentTypePanel.add(JLabel7);
                JLabel7.setForeground(java.awt.Color.black);
                JLabel7.setBounds(50,190,170,22);

        JLabel8.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel8.setText("Existing Component Types");
                componentTypePanel.add(JLabel8);
                JLabel8.setForeground(java.awt.Color.black);
                JLabel8.setBounds(50,150,170,22);
                compAddButton.setText("Add");
                compAddButton.setActionCommand("jbutton");
                componentTypePanel.add(compAddButton);
                compAddButton.setBounds(12,297,75,22);
                compDeleteButton.setText("Delete");
                compDeleteButton.setActionCommand("jbutton");
                componentTypePanel.add(compDeleteButton);
                compDeleteButton.setBounds(168,297,75,22);
                compClearButton.setText("Clear");
                compClearButton.setActionCommand("jbutton");
                componentTypePanel.add(compClearButton);
                compClearButton.setBounds(246,297,75,22);
                compEditButton.setText("Edit");
                compEditButton.setActionCommand("jbutton");
                componentTypePanel.add(compEditButton);
                compEditButton.setBounds(90,297,75,22);
```

```
componentTypePanel.add(compIDTextField);
compIDTextField.setBounds(225,70,300,22);
componentTypePanel.add(compNameTextField);
compNameTextField.setBounds(225,110,300,22);
componentTypePanel.add(existedCompComboBox);
existedCompComboBox.setBounds(225,150,300,22);
compDescriptionTextArea.setLineWrap(true);
compDescriptionTextArea.setWrapStyleWord(true);
componentTypePanel.add(compDescriptionTextArea);
compDescriptionTextArea.setBounds(225,190,300,80);
compSaveButton.setText("Save");
compSaveButton.setActionCommand("Save");
componentTypePanel.add(compSaveButton);
compSaveButton.setBounds(492,297,75,22);

    JLabel17.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
JLabel17.setText("Project Label:");
componentTypePanel.add(JLabel17);
JLabel17.setForeground(java.awt.Color.black);
JLabel17.setFont(new Font("Dialog", Font.BOLD, 18));
JLabel17.setBounds(0,6,280,24);
compLabel.setText("Label");
componentTypePanel.add(compLabel);
compLabel.setForeground(java.awt.Color.black);
compLabel.setFont(new Font("Dialog", Font.BOLD, 18));
compLabel.setBounds(294,6,280,24);
EHLPanel.setDoubleBuffered(false);
EHLPanel.setLayout(null);
configManagTabbedPane.add(EHLPanel);
EHLPanel.setBounds(2,27,575,330);
EHLPanel.setVisible(false);

    JLabel13.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
JLabel13.setText("Evolution Process Name");
EHLPanel.add(JLabel13);
JLabel13.setForeground(java.awt.Color.black);
JLabel13.setBounds(60,70,150,22);

    JLabel14.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
JLabel14.setText("Evolution Process");
EHLPanel.add(JLabel14);
JLabel14.setForeground(java.awt.Color.black);
JLabel14.setBounds(60,210,150,22);
EHLAddButton.setText("Add");
EHLAddButton.setActionCommand("jbutton");
EHLPanel.add(EHLAddButton);
EHLAddButton.setBounds(10,297,75,22);
EHLDeleteButton.setText("Delete");
EHLDeleteButton.setActionCommand("jbutton");
EHLPanel.add(EHLDeleteButton);
EHLDeleteButton.setBounds(166,297,75,22);
EHLClearButton.setText("Clear");
EHLClearButton.setActionCommand("jbutton");
EHLPanel.add(EHLClearButton);
EHLClearButton.setBounds(244,297,75,22);
EHLEditButton.setText("Edit");
EHLEditButton.setActionCommand("jbutton");
EHLPanel.add(EHLEditButton);
EHLEditButton.setBounds(88,297,75,22);
EHLPanel.add(EHLNameTextField);
EHLNameTextField.setBounds(215,70,300,22);
EHLPathTextField.setDoubleBuffered(true);
EHLPathTextField.setEditable(false);
EHLPanel.add(EHLPathTextField);
EHLPathTextField.setBackground(java.awt.Color.white);
EHLPathTextField.setForeground(java.awt.Color.black);
EHLPathTextField.setBounds(215,210,300,22);
EHLDoneButton.setText("Done");
EHLDoneButton.setActionCommand("Done");
EHLPanel.add(EHLDoneButton);
EHLDoneButton.setBounds(493,297,75,22);
```

```
        label.setHorizontalAlignment(com.sun.java.swing.SwingConstants
.RIGHT);
                label.setText("Project Label:");
                EHLPanel.add(label);
                label.setForeground(java.awt.Color.black);
                label.setFont(new Font("Dialog", Font.BOLD, 18));
                label.setBounds(2,6,280,24);
                projectLabel.setText("Label");
                EHLPanel.add(projectLabel);
                projectLabel.setForeground(java.awt.Color.black);
                projectLabel.setFont(new Font("Dialog", Font.BOLD,
18));
                projectLabel.setBounds(294,6,280,24);

        JLabel15.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel15.setText("Existing Evolution Process");
                EHLPanel.add(JLabel15);
                JLabel15.setForeground(java.awt.Color.black);
                JLabel15.setBounds(60,250,150,22);
                EHLPanel.add(existedEHLComboBox);
                existedEHLComboBox.setBounds(215,250,300,22);

        JLabel20.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel20.setText("Step Types");
                JLabel20.setNextFocusableComponent(compAddButton);
                EHLPanel.add(JLabel20);
                JLabel20.setForeground(java.awt.Color.black);
                JLabel20.setBounds(60,110,150,22);
                stepTypesScrollPane.setOpaque(true);
                EHLPanel.add(stepTypesScrollPane);
                stepTypesScrollPane.setBounds(215,110,300,80);
                stepTypesScrollPane.getViewport().add(stepTypesList);
                stepTypesList.setBounds(0,0,297,77);
                dependencyPanel.setLayout(null);
                configManagTabbedPane.add(dependencyPanel);
                dependencyPanel.setBounds(2,27,575,330);
                dependencyPanel.setVisible(false);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Step Types");
                dependencyPanel.add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(30,110,240,22);

        JLabel10.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel10.setText("Output Component Type");
                dependencyPanel.add(JLabel10);
                JLabel10.setForeground(java.awt.Color.black);
                JLabel10.setBounds(30,150,240,22);

        JLabel12.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel12.setText("Primary Input Component Type");
                dependencyPanel.add(JLabel12);
                JLabel12.setForeground(java.awt.Color.black);
                JLabel12.setBounds(30,190,240,22);
                depenOKButton.setText("OK");
                depenOKButton.setActionCommand("jbutton");
                dependencyPanel.add(depenOKButton);
                depenOKButton.setBounds(390,290,75,22);
                depenCancelButton.setText("Cancel");
                depenCancelButton.setActionCommand("jbutton");
                dependencyPanel.add(depenCancelButton);
                depenCancelButton.setBounds(470,290,75,22);
                depenSecondaryTextField.setEditable(false);
                dependencyPanel.add(depenSecondaryTextField);

        depenSecondaryTextField.setBackground(java.awt.Color.white);
                depenSecondaryTextField.setBounds(275,230,270,22);
```

```
                depenOutputTextField.setEditable(false);
                dependencyPanel.add(depenOutputTextField);

        depenOutputTextField.setBackground(java.awt.Color.white);

        depenOutputTextField.setForeground(java.awt.Color.black);
                depenOutputTextField.setFont(new Font("SansSerif",
Font.PLAIN, 12));
                depenOutputTextField.setBounds(275,150,270,22);
                dependencyPanel.add(depenStepComboBox);
                depenStepComboBox.setBounds(275,110,270,22);
                depenPrimaryTextField.setEditable(false);
                dependencyPanel.add(depenPrimaryTextField);

        depenPrimaryTextField.setBackground(java.awt.Color.white);

        depenPrimaryTextField.setForeground(java.awt.Color.black);
                depenPrimaryTextField.setBounds(275,190,270,22);

        JLabel18.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel18.setText("Project Label:");
                dependencyPanel.add(JLabel18);
                JLabel18.setForeground(java.awt.Color.black);
                JLabel18.setFont(new Font("Dialog", Font.BOLD, 18));
                JLabel18.setBounds(0,6,280,24);
                depLabel.setText("Label");
                dependencyPanel.add(depLabel);
                depLabel.setForeground(java.awt.Color.black);
                depLabel.setFont(new Font("Dialog", Font.BOLD, 18));
                depLabel.setBounds(294,6,280,24);

        JLabel19.setHorizontalTextPosition(com.sun.java.swing.SwingCo
nstants.CENTER);

        JLabel19.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
```



```
                JLabel19.setText("Evolution Process");
                dependencyPanel.add(JLabel19);
                JLabel19.setForeground(java.awt.Color.black);
                JLabel19.setBounds(30,70,240,22);
                dependencyPanel.add(depenEHLComboBox);
                depenEHLComboBox.setBounds(275,70,270,22);
                secondaryButton.setText("Secondary Input Component
Type(s)");
                secondaryButton.setActionCommand("Secondary Input
Component Type(s)");
                dependencyPanel.add(secondaryButton);
                secondaryButton.setBounds(30,230,240,22);
                configManagTabbedPane.setSelectedIndex(0);

        configManagTabbedPane.setSelectedComponent(stepTypePanel);
                try {
                        configManagTabbedPane.setTitleAt(0,"Step
Type");
                }
                catch(ArrayIndexOutOfBoundsException e) {
        System.out.println(e);
                }

        configManagTabbedPane.setTitleAt(1,"Component Type");

        configManagTabbedPane.setTitleAt(2,"Evolution Process");

        configManagTabbedPane.setTitleAt(3,"Dependency");
                   doneButton.setText("Finish");
                   doneButton.setActionCommand("OK");
                   getContentPane().add(doneButton);
                   doneButton.setBounds(262,390,75,22);
                //}}

                //{{INIT_MENUS
                //}}
```

```
        //{{REGISTER_LISTENERS
        SymAction lSymAction = new SymAction();
        stepAddButton.addActionListener(lSymAction);
        stepEditButton.addActionListener(lSymAction);
        stepDeleteButton.addActionListener(lSymAction);
        stepClearButton.addActionListener(lSymAction);
        compAddButton.addActionListener(lSymAction);
        compEditButton.addActionListener(lSymAction);
        compDeleteButton.addActionListener(lSymAction);
        compClearButton.addActionListener(lSymAction);
        EHLAddButton.addActionListener(lSymAction);
        EHLEditButton.addActionListener(lSymAction);
        EHLDeleteButton.addActionListener(lSymAction);
        EHLClearButton.addActionListener(lSymAction);
        depenOKButton.addActionListener(lSymAction);
        depenCancelButton.addActionListener(lSymAction);
        doneButton.addActionListener(lSymAction);
        EHLDoneButton.addActionListener(lSymAction);
        stepSaveButton.addActionListener(lSymAction);
        compSaveButton.addActionListener(lSymAction);
        SymItem lSymItem = new SymItem();
        depenStepComboBox.addItemListener(lSymItem);
        depenEHLComboBox.addItemListener(lSymItem);
        existedStepComboBox.addItemListener(lSymItem);
        existedCompComboBox.addItemListener(lSymItem);
        existedEHLComboBox.addItemListener(lSymItem);
        SymMouse aSymMouse = new SymMouse();
        stepTypesList.addMouseListener(aSymMouse);
        secondaryButton.addActionListener(lSymAction);
        //}}

        stepTypesList.setModel( listModel );
        setEnabled(3, false);
    }

/**
     * Is launched when a user wants to create or edit step.cfg,
component.cfg, loop.cfg, or dependency.cfg
     *
     * @param selectedProject : current project name
     * @param pathName : the path of the current project
     */
    public ProjectSchemaFrame( String selectedProject, String pathName ){
        this();
        this.setProjectLabel( selectedProject );
        this.pathName = pathName;
        this.readInputFiles( this.pathName );
    }

    public void setVisible(boolean b)
        {
            if (b)
                setLocation(50, 50);
            super.setVisible(b);
        }

        static public void main(String args[])
        {
            (new ProjectSchemaFrame()).setVisible(true);
        }

        public void addNotify()
        {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                return;
            frameSizeAdjusted = true;
```

```
                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JPanel mainPanel = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JTabbedPane configManagTabbedPane = new
com.sun.java.swing.JTabbedPane();
        com.sun.java.swing.JPanel stepTypePanel = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel4 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton stepAddButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton stepDeleteButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton stepClearButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton stepEditButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField stepIDTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField stepNameTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox existedStepComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextArea stepDescriptionTextArea = new
com.sun.java.swing.JTextArea();
        com.sun.java.swing.JButton stepSaveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel16 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel stepLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JPanel componentTypePanel = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JLabel JLabel5 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel6 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel7 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel8 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton compAddButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton compDeleteButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton compClearButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton compEditButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField compIDTextField = new
com.sun.java.swing.JTextField();
```

```
        com.sun.java.swing.JTextField compNameTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox existedCompComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextArea compDescriptionTextArea = new
com.sun.java.swing.JTextArea();
        com.sun.java.swing.JButton compSaveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel17 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel compLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JPanel EHLPanel = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JLabel JLabel13 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel14 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton EHLAddButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton EHLDeleteButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton EHLClearButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton EHLEditButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField EHLNameTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField EHLPathTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JButton EHLDoneButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel label = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel projectLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel15 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox existedEHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel20 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane stepTypesScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JList stepTypesList = new
com.sun.java.swing.JList();
        com.sun.java.swing.JPanel dependencyPanel = new
com.sun.java.swing.JPanel();
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel10 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel12 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton depenOKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton depenCancelButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField depenSecondaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField depenOutputTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JComboBox depenStepComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JTextField depenPrimaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel18 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel depLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel19 = new
com.sun.java.swing.JLabel();
```

```
        com.sun.java.swing.JComboBox depenEHLComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JButton secondaryButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton doneButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == stepAddButton)

        stepAddButton_actionPerformed(event);
                        else if (object == stepEditButton)

        stepEditButton_actionPerformed(event);
                        else if (object == stepDeleteButton)

        stepDeleteButton_actionPerformed(event);
                        else if (object == stepClearButton)

        stepClearButton_actionPerformed(event);
                        else if (object == compAddButton)

        compAddButton_actionPerformed(event);
                        else if (object == compEditButton)

        compEditButton_actionPerformed(event);
                        else if (object == compDeleteButton)

compDeleteButton_actionPerformed(event);
                        else if (object == compClearButton)

compClearButton_actionPerformed(event);
                        else if (object == EHLAddButton)

EHLAddButton_actionPerformed(event);
                        else if (object == EHLEditButton)

EHLEditButton_actionPerformed(event);
                        else if (object == EHLDeleteButton)

EHLDeleteButton_actionPerformed(event);
                        else if (object == EHLClearButton)

EHLClearButton_actionPerformed(event);
                        else if (object == depenOKButton)

depenOKButton_actionPerformed(event);
                        else if (object == depenCancelButton)

depenCancelButton_actionPerformed(event);
                        else if (object == doneButton)
                                doneButton_actionPerformed(event);
                        else if (object == EHLDoneButton)

EHLDoneButton_actionPerformed(event);
                        else if (object == stepSaveButton)

stepSaveButton_actionPerformed(event);
                        if (object == compSaveButton)

compSaveButton_actionPerformed(event);
                        if (object == secondaryButton)

secondaryButton_actionPerformed(event);
```

```
                }
        }

    /**
     * Set the current project name for each panel's title
     *
     * @param selectedProject : the current project name
     */
    public void setProjectLabel( String selectedProject ){
        this.projectLabel.setText( selectedProject );
        this.stepLabel.setText( selectedProject );
        this.compLabel.setText( selectedProject );
        this.depLabel.setText( selectedProject );
    }

    /**
     * Add new step type object into combo box and stepVector
     */
            public void
stepAddButton_actionPerformed(java.awt.event.ActionEvent event)
            {
        String stepID = stepIDTextField.getText();
        String stepName = stepNameTextField.getText();
        String stepDescription = stepDescriptionTextArea.getText();
            StepType stepType = new StepType( stepID, stepName,
stepDescription );

            this.stepVector.addElement( stepType );
            this.setListModel( this.stepVector );

              this.setStepComboBox( this.stepVector );
              saveTab[0] = false;

              //Clean up the frame
              this.stepClearButton_actionPerformed( null );
            }

    /**
     * Edit selected step type object and delete before add it into combo box
and stepVector
     */
            public void
stepEditButton_actionPerformed(java.awt.event.ActionEvent event)
            {
              if( this.selectedStepIndex > 0 ){
          String stepID = stepIDTextField.getText();
          String stepName = stepNameTextField.getText();
          String stepDescription = stepDescriptionTextArea.getText();
              StepType stepType = new StepType( stepID, stepName,
stepDescription );

              this.stepVector.setElementAt( stepType, this.selectedStepIndex-
1 );
              this.setListModel( this.stepVector );
                this.setStepComboBox( this.stepVector );
              saveTab[0] = false;
        }

              //Clean up the frame
              this.stepClearButton_actionPerformed( null );
    }

    /**
     * Delete selected step type object and remove it from combo box and
stepVector
     */
            public void
stepDeleteButton_actionPerformed(java.awt.event.ActionEvent event)
            {
              if( this.selectedStepIndex > 0 ){
                this.stepVector.removeElementAt( this.selectedStepIndex - 1
);
            this.setListModel( this.stepVector );
                this.setStepComboBox( this.stepVector );
```

```
            saveTab[0] = false;
        }

            //Clean up the frame
            this.stepClearButton_actionPerformed( null );
          }

    /**
     * Refresh all text fields in step panel
     */
          public void
stepClearButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            this.stepIDTextField.setText("");
            this.stepNameTextField.setText("");
            this.stepDescriptionTextArea.setText("");
            if( this.existedStepComboBox.getItemCount() > 0 ){
              this.existedStepComboBox.setSelectedIndex(0);
            }
          }

    /**
     * Save all step type objects in stepVector in step.cfg file
     */
          public void
stepSaveButton_actionPerformed(java.awt.event.ActionEvent event)
          {
       try{
          FileOutputStream stepFileOut = new FileOutputStream(
this.pathName+"\\step.cfg" );
          ObjectOutputStream stepOut= new ObjectOutputStream(
stepFileOut );

                if( this.stepVector.size()> 0 ){
                  if(stepOut != null ){
                    stepOut.writeObject( this.stepVector );
                    this.setStepComboBox( this.stepVector );
                    saveTab[0] = true;
                  }
                    stepOut.flush();
                    stepOut.close();
                    stepFileOut.close();
                }
          }
          catch( FileNotFoundException fe ){
            debug("FileNotFoundException: "+fe);
          }
          catch( IOException e ){
            debug("IOException: "+e);
          }

          //Clean up the frame
              this.stepClearButton_actionPerformed( null );
            }

    /**
     * Add new component type object into combo box and compVector
     */
          public void
compAddButton_actionPerformed(java.awt.event.ActionEvent event)
          {
          String compID = compIDTextField.getText();
          String compName = compNameTextField.getText();
          String compDescription = compDescriptionTextArea.getText();

          ComponentType compType = new ComponentType( compID,
compName, compDescription );
          this.compVector.addElement( compType );

            this.setCompComboBox( this.compVector );
            saveTab[1] = false;

            //Clean up the frame
            this.stepClearButton_actionPerformed( null );
```

```
    }

    /**
     * Edit selected component type object and delete before add it into
combo box and compVector
     */
          public void
compEditButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            if( this.selectedCompIndex > 0 ){
          String compID = compIDTextField.getText();
          String compName = compNameTextField.getText();
          String compDescription = compDescriptionTextArea.getText();

            ComponentType compType = new ComponentType( compID,
compName, compDescription );
            this.compVector.setElementAt( compType,
this.selectedCompIndex-1 );

              this.setCompComboBox( this.compVector );
            saveTab[1] = false;
       }

            //Clean up the frame
            this.compClearButton_actionPerformed( null );
          }

    /**
     * Delete selected component type object and remove it from combo box
and compVector
     */
          public void
compDeleteButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            if( this.selectedCompIndex > 0 ){
              this.compVector.removeElementAt( this.selectedCompIndex
- 1 );
              this.setCompComboBox( this.compVector );
            saveTab[1] = false;
        }

            //Clean up the frame
            this.compClearButton_actionPerformed( null );
          }

    /**
     * Refresh all text fields in component panel
     */
          public void
compClearButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            this.compIDTextField.setText("");
            this.compNameTextField.setText("");
            this.compDescriptionTextArea.setText("");
            if( this.existedCompComboBox.getItemCount() > 0 ){
              this.existedCompComboBox.setSelectedIndex(0);
            }
          }

    /**
     * Save all component type objects in compVector in component.cfg file
     */
          public void
compSaveButton_actionPerformed(java.awt.event.ActionEvent event)
          {
        try{
          FileOutputStream compFileOut = new FileOutputStream(
this.pathName+"\\component.cfg");
          ObjectOutputStream compOut= new ObjectOutputStream(
compFileOut );
              if( this.compVector.size()> 0 ){
                if(compOut != null ){
                  compOut.writeObject(this.compVector);
                    saveTab[1] = true;
```

```
            this.setCompComboBox( this.compVector );
                }
                compOut.flush();
                compOut.close();
                compFileOut.close();
              }
        }
        catch( FileNotFoundException fe ){
          debug("FileNotFoundException: "+fe);
        }
        catch( IOException e ){
          debug("IOException: "+e);
        }
        //Clear the Frame
            this.compClearButton_actionPerformed( null );
          }

    /**
     * Add new EHL object into combo box and EHLVector
     */
          public void
EHLAddButton_actionPerformed(java.awt.event.ActionEvent event)
          {
        String EHLName = EHLNameTextField.getText();
        String EHLPath = EHLPathTextField.getText();

        StringTokenizer st = new StringTokenizer( EHLPath, "," );
            Vector tokenizeVector = new Vector();
            while( st.hasMoreTokens() ){
              tokenizeVector.addElement( st.nextToken() );
            }

        if( tokenizeVector.size()< 2 ){
          JOptionPane.showMessageDialog(this, "The loop must have at least
2 steps!",
                        "Error", JOptionPane.ERROR_MESSAGE);
          return;
        }
        else{
            EHL ehl = new EHL( EHLName, EHLPath );
            this.EHLVector.addElement( ehl );
              this.setEHLComboBox( this.EHLVector );
            saveTab[2] = false;

              //Clean up the frame
              this.EHLClearButton_actionPerformed( null );
        }
          }

    /**
     * Edit selected EHL object and delete before add it into combo box and
EHLVector
     */
          public void
EHLEditButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            if( this.selectedEHLIndex > 0 ){
          String EHLName = EHLNameTextField.getText();
          String EHLPath = EHLPathTextField.getText();
            EHL ehl = new EHL( EHLName, EHLPath );
            this.EHLVector.setElementAt( ehl, this.selectedEHLIndex-1 );
              this.setEHLComboBox( this.EHLVector );
            saveTab[2] = false;
        }

            //Clean up the frame
            this.EHLClearButton_actionPerformed( null );
          }

    /**
     * Delete selected EHL object and remove it from combo box and
EHLVector
     */
```

```
        public void
EHLDeleteButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          if( this.selectedEHLIndex > 0 ){
            this.EHLVector.removeElementAt( this.selectedEHLIndex - 1
);
            this.setEHLComboBox( this.EHLVector );
          saveTab[2] = false;
     }

          //Clean up the frame
          this.EHLClearButton_actionPerformed( null );
        }

  /**
   * Refresh all text fields in EHL panel
   */
        public void
EHLClearButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          this.EHLNameTextField.setText("");
          this.EHLPathTextField.setText("");
        }

  /**
   * Save all EHL objects in EHLVector in loop.cfg file and go directly
into dependency panel
   */
        public void
EHLDoneButton_actionPerformed(java.awt.event.ActionEvent event)
        {
     try{
       FileOutputStream EHLFileOut = new FileOutputStream(
this.pathName+"\\loop.cfg" );
        ObjectOutputStream EHLOut = new ObjectOutputStream(
EHLFileOut );
          if( this.EHLVector.size()> 0 ){
            if(EHLOut != null ){
            EHLOut.writeObject( this.EHLVector );
         saveTab[2] = true;
          this.readDepFile();
               this.setDepenEHLComboBox( this.EHLVector);

               //Only use for EHL tabbedPane
               this.setEnabled( 3, true);
               this.configManagTabbedPane.setSelectedIndex( 3 );

               this.setEnabled( 0, false );
               this.setEnabled( 1, false );
               this.setEnabled( 2, false );
             }
        EHLOut.flush();
        EHLOut.close();
        EHLFileOut.close();
           }
    }
    catch( FileNotFoundException fe ){
      debug("FileNotFoundException: "+fe);
    }
    catch( IOException e ){
      debug("IOException: "+e);
    }

    //Clean up the frame
          this.EHLClearButton_actionPerformed( null );
       }

 /**
   * Save all dependency objects in depenHashtable in dependency.cfg file
   * Before saving it, check all secondary inputs of each dependency to be
set or not.
   * The warning message will show up if one of dependency objects
   * didn't set the secondary input
```

```
    */
        public void
depenOKButton_actionPerformed(java.awt.event.ActionEvent event)
        {
      saveLastDep();

          Vector v = new Vector();
          Vector v2 = new Vector();
      Dependency dep = null;

      Enumeration enum = (Enumeration)listModel.elements();
      while( enum.hasMoreElements() ){
        if(
!this.depenHashtable.containsKey((String)enum.nextElement())){
            v.addElement("");
            v2.addElement("");
        }
      }
      enum = null;
      try{
        FileOutputStream depFileOut = new FileOutputStream(
this.pathName+"\\dependency.cfg" );
        ObjectOutputStream dependencyOut = new ObjectOutputStream(
depFileOut );
        if( this.depenHashtable.size() > 0 ){
            enum = this.depenHashtable.elements();
            while( enum.hasMoreElements() ){
              Dependency d = (Dependency)enum.nextElement();
          if( d!= null ){
            v.addElement(((String)d.getSecondaryInput()).trim());
              v2.addElement(((String)d.getStep()).trim());
            }
            else{
              v.addElement("");
              v2.addElement("");
            }
            }
                for( int i=0; i<v.size(); i++ ){
                  if( ((String)v.elementAt(i)).equals("") ){
                    if(
!((String)v2.elementAt(i)).equals(STEP_TYPE_TITLE) ){
                JOptionPane.showMessageDialog(this, "Can not process
since \nsome steps do not have the secondary \n input component types!",
                                "Error",
JOptionPane.ERROR_MESSAGE);
                return;
              }
                }
              }
        if( dependencyOut != null ){
            dependencyOut.writeObject( this.depenHashtable );
          saveTab[3] = true;
            }
            dependencyOut.flush();
            dependencyOut.close();
            depFileOut.close();
          }
      }
      catch( FileNotFoundException fe ){
        debug("FileNotFoundException_DepFileOut: "+fe);
      }
      catch( IOException e ){
        debug("IOException: "+e);
      }

        this.setEnabled( 3, false);
        this.configManagTabbedPane.setSelectedIndex( 0 );
        this.setEnabled( 0, true );
        this.setEnabled( 1, true );
        this.setEnabled( 2, true );
      this.depenPrimaryTextField.setText("");
      this.depenOutputTextField.setText("");
      this.depenSecondaryTextField.setText("");
```

```
        }

    /**
     * Before leave the dependency panel, save all the depency objects in
depenHashtable
     */
          public void
depenCancelButton_actionPerformed(java.awt.event.ActionEvent event)
          {
       try{
         FileOutputStream depFileOut = new FileOutputStream(
this.pathName+"\\dependency.cfg" );
         ObjectOutputStream dependencyOut = new ObjectOutputStream(
depFileOut );
                if( this.depenHashtable.size() > 0 ){
           if( dependencyOut != null ){
                    dependencyOut.writeObject( this.depenHashtable );
                  }
                  dependencyOut.flush();
                  dependencyOut.close();
                  depFileOut.close();
                }
      }
      catch(IOException e ){
        debug("IOException_DepCancel: "+e);
      }

            this.setEnabled( 3, false);

            this.configManagTabbedPane.setSelectedIndex( 0 );

            this.setEnabled( 0, true );
            this.setEnabled( 1, true );
            this.setEnabled( 2, true );

      this.depenPrimaryTextField.setText("");
      this.depenOutputTextField.setText("");
      this.depenSecondaryTextField.setText("");
          }

    /**
     * Check all the changes had saved or not before exit
ProjectSchemaFrame
     */
          public void
doneButton_actionPerformed(java.awt.event.ActionEvent event)
          {
       for( int i=0; i<saveTab.length; i++ ){
         if( !saveTab[i] ){
               Object[] options = { "Yes", "No" };
           int j = JOptionPane.showOptionDialog(this, "Would you like to
save?",
           "Warning", JOptionPane.DEFAULT_OPTION,
JOptionPane.WARNING_MESSAGE, null, options, options[0]);
           if( j==0 ){
             stepSaveButton_actionPerformed(event);
             compSaveButton_actionPerformed(event);
             EHLDoneButton_actionPerformed(event);
             depenOKButton_actionPerformed(event);

           }
         }
       }
             setVisible( false );
             dispose();
           }

    /**
     * Set enable a tab panel
     *
     * @param tabIndex : an index of panel
     * @param flag : true or false
     */
          public void setEnabled( int tabIndex, boolean flag ){
```

```
            this.configManagTabbedPane.setEnabledAt( tabIndex, flag );
        }

    /**
     * Short cut to print the output
     * @param string : the output string
     */
    public void debug( String string ){
       System.out.println(string);
    }

    class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == depenStepComboBox)

        depenStepComboBox_itemStateChanged(event);
                        else if (object == depenEHLComboBox)

        depenEHLComboBox_itemStateChanged(event);
                        else if (object == existedStepComboBox)

        existedStepComboBox_itemStateChanged(event);
                        else if (object == existedCompComboBox)

        existedCompComboBox_itemStateChanged(event);
                        else if (object == existedEHLComboBox)

        existedEHLComboBox_itemStateChanged(event);
                }
        }

    /**
     * Refresh stepHashtable and existedStepComboBox, and add new items
     *
     * @param stepVector : contains the current step type objects
     */
    public void setStepComboBox( Vector stepVector ){
       this.existedStepComboBox.removeAllItems();
       this.existedStepComboBox.addItem(STEP_TYPE_TITLE);

       this.stepHashtable = new Hashtable();

       for( int i=0; i< stepVector.size(); i++ ){
          StepType st = (StepType)stepVector.elementAt( i );
          String stID = st.getStepID();

          //set step Hashtable
          this.stepHashtable.put( stID, st );

          this.existedStepComboBox.addItem(stID);
       }
    }

    /**
     * Refresh compHashtable and  existedCompComboBox, and add new
items
     *
     * @param compVector : contains the current component type objects
     */
    public void setCompComboBox( Vector compVector ){
       this.existedCompComboBox.removeAllItems();
       this.existedCompComboBox.addItem(COMPONENT_TYPE_TITLE);

       this.compHashtable = new Hashtable();

       for( int i=0; i< compVector.size(); i++ ){
          ComponentType ct = (ComponentType)compVector.elementAt( i );
          String ctID = ct.getComponentID();

          //set step Hashtable
```

```
            this.compHashtable.put( ctID, ct );

            this.existedCompComboBox.addItem(ctID);
        }
    }

    /**
     * Refresh EHLHashtable and existedEHLComboBox, and add new
items
     *
     * @param EHLVector : contains the current EHL objects
     */
    public void setEHLComboBox( Vector EHLVector ){
        this.existedEHLComboBox.removeAllItems();
        this.existedEHLComboBox.addItem(PROCESS_TITLE);

        this.EHLHashtable = new Hashtable();

        for( int i=0; i< EHLVector.size(); i++ ){
            EHL ehl = (EHL)EHLVector.elementAt( i );
            String EHLName = ehl.getEHLName();

            //set step Hashtable
            this.EHLHashtable.put( EHLName, ehl );

            this.existedEHLComboBox.addItem(EHLName);
        }
    }

    /**
     * Refresh depenEHLComboBox and add new items
     *
     * @param EHLVector : contains the current EHL objects
     */
    public void setDepenEHLComboBox( Vector EHLVector ){
        this.depenEHLComboBox.removeAllItems();

        for( int j=0; j< EHLVector.size(); j++ ){
            EHL ehl = (EHL)EHLVector.elementAt(j);
            this.depenEHLComboBox.addItem(ehl.getEHLName());
        }
    }

    /**
     * Refresh dependStepComboBox, add new items,
     * and create new directory for the selected step
     *
     * @param stepVector : contains the current step names
     */
    public void setDependStepComboBox( Vector stepVector ){
        String loopName =
(String)this.depenEHLComboBox.getSelectedItem();

        this.dependStepComboBox.removeAllItems();
        this.dependStepComboBox.addItem(STEP_TYPE_TITLE);
        for( int i=0; i< stepVector.size(); i++ ){
            String depStep = ((String)stepVector.elementAt( i )).trim();
            this.dependStepComboBox.addItem(depStep);
            File directory = new File( this.pathName, depStep );
            if( !directory.isDirectory() ){
                directory.mkdir();
            }
        }
    }

    /**
     * View the selected step type object
     *
     * @param st : selected step type object
     */
    public void setStepInfo( StepType st ){
        stepIDTextField.setText( st.getStepID() );
        stepNameTextField.setText( st.getStepName() );
        stepDescriptionTextArea.setText( st.getStepDescription() );
```

```
}

/**
 * View the selected component type object
 *
 * @param ct : selected component type object
 */
public void setCompInfo( ComponentType ct ){
   compIDTextField.setText( ct.getComponentID() );
   compNameTextField.setText( ct.getComponentName() );
   compDescriptionTextArea.setText( ct.getComponentDescription() );
}

/**
 * View the selected EHL object
 *
 * @param ehl : selected EHL object
 */
public void setEHLInfo( EHL ehl ){
   EHLNameTextField.setText( ehl.getEHLName() );
   EHLPathTextField.setText( ehl.getEHLPath() );
}

/**
 * View the selected dependency object
 *
 * @param selectedDepStep : selected dependency object name
 */
public void setDepInfo( String selectedDepStep ){
   if( !selectedDepStep.equals(STEP_TYPE_TITLE) ){
      StringTokenizer st = new StringTokenizer( selectedDepStep,"-");
            String before_ = st.nextToken("-");
            String _after = st.nextToken("-");
      this.depenPrimaryTextField.setText( _after );
      this.depenOutputTextField.setText( _after);

            if( this.depenHashtable.containsKey( selectedDepStep ) ){
            Dependency dep = (Dependency)this.depenHashtable.get(
selectedDepStep );
            String secondInput = dep.getSecondaryInput();
            if( !secondInput.equals("") ){
               this.depenSecondaryTextField.setText(secondInput);
                  }
         }
      }
   }

   /**
    * Read step.cfg, component.cfg, and loop.cfg files and insert their
contents
    * into stepVector, compVector, and EHLVector respectively
    *
    * @param pathName : the path of current project
    */
   public void readInputFiles( String pathName ){
      File stepFile = new File(this.pathName, "step.cfg");
      if( stepFile.exists() ){
         try{
         FileInputStream fileInput = new FileInputStream( stepFile );
         ObjectInputStream stepIn = new ObjectInputStream( fileInput );
            if( stepIn != null ){
               this.stepVector = (Vector)stepIn.readObject();
               if( this.stepVector.size() > 0 ){
                  this.setStepComboBox( this.stepVector );
                  this.setListModel(this.stepVector);
               }
            }
            stepIn.close();
            fileInput.close();
         }
         catch( IOException e ){
            debug("IOException_StepInitial: "+e);
            if( this.stepVector.size() > 0 ){
               this.setStepComboBox( this.stepVector );
```

```
            this.setListModel(this.stepVector);
          }
        }
        catch( ClassNotFoundException ex ){
          debug("ClassNotFoundException_SteppInitial: "+ex);
          if( this.stepVector.size() > 0 ){
            this.setStepComboBox( this.stepVector );
            this.setListModel(this.stepVector);
          }
        }
      }

      File componentFile = new File(this.pathName, "component.cfg");
      if( componentFile.exists() ){
        try{
          FileInputStream fileInput = new FileInputStream( componentFile)
;
          ObjectInputStream compIn = new ObjectInputStream( fileInput );
          if( compIn != null ){
            this.compVector = (Vector)compIn.readObject();
            if( this.compVector.size() > 0 ){
              this.setCompComboBox( this.compVector );
            }
          }
          compIn.close();
          fileInput.close();
        }
        catch( IOException e ){
          debug("IOException_CompInitial: "+e);
          if( this.compVector.size() > 0 ){
            this.setCompComboBox( this.compVector );
          }
        }
        catch( ClassNotFoundException ex ){
          debug("ClassNotFoundException_CompInitial: "+ex);
          if( this.compVector.size() > 0 ){
            this.setCompComboBox( this.compVector );
          }
        }
      }

      File loopFile = new File(this.pathName, "loop.cfg");
      if( loopFile.exists() ){
        try{
          FileInputStream fileInput = new FileInputStream( loopFile );
          ObjectInputStream EHLIn = new ObjectInputStream( fileInput );
          if( EHLIn != null ){
            this.EHLVector = (Vector)EHLIn.readObject();
            if( this.EHLVector.size() > 0 ){
              this.setEHLComboBox( this.EHLVector );
            }
          }
          EHLIn.close();
          fileInput.close();
        }
        catch( IOException e ){
          debug("IOException_EHLInInitial: "+e);
          if( this.EHLVector.size() > 0 ){
            this.setEHLComboBox( this.EHLVector );
          }
        }
        catch( ClassNotFoundException ex ){
          debug("ClassNotFoundException_EHLInInitial: "+ex);
          if( this.EHLVector.size() > 0 ){
            this.setEHLComboBox( this.EHLVector );
          }
        }
      }
    }

    /**
     * Read dependency.cfg file and insert its content into depenHashtable
     */
    public void readDepFile(){
```

```
        File dependencyFile = new File(this.pathName, "dependency.cfg");
        if( dependencyFile.exists() ){
          try{
            FileInputStream fileInput = new FileInputStream( dependencyFile
);
            ObjectInputStream dependencyIn = new ObjectInputStream(
fileInput );
            if( dependencyIn != null ){
              this.depenHashtable = (Hashtable)dependencyIn.readObject();
            }
            dependencyIn.close();
            fileInput.close();
          }
          catch( IOException e ){
            debug("IOException_readDepIn: "+e);
          }
          catch( ClassNotFoundException ex ){
            debug("ClassNotFoundException_readDepIn: "+ex);
          }
        }
      }
    /**
      * List all existing step type objects in the project
      * and allow a user to view or edit them
      */
          void
existedStepComboBox_itemStateChanged(java.awt.event.ItemEvent event)
          {
        if( event.getStateChange() == ItemEvent.SELECTED ){
          String selectedStepID = (String)event.getItem();
          this.selectedStepIndex =
this.existedStepComboBox.getSelectedIndex();
          if( this.selectedStepIndex == 0 ){
            //Clean up the frame
                this.stepClearButton_actionPerformed( null );
              }
          else if( this.selectedStepIndex > 0 ){
            if( this.stepHashtable.containsKey( selectedStepID ) ){
              this.setStepInfo( (StepType)this.stepHashtable.get(
selectedStepID ) );
            }
          }
        }
          }

    /**
      * List all existing component type objects in the project
      * and allow a user to view or edit them
      */
          void
existedCompComboBox_itemStateChanged(java.awt.event.ItemEvent
event)
          {
        if( event.getStateChange() == ItemEvent.SELECTED ){
          String selectedCompID = (String)event.getItem();
          this.selectedCompIndex =
this.existedCompComboBox.getSelectedIndex();
          if( this.selectedCompIndex == 0 ){
            //Clean up the frame
                  this.compClearButton_actionPerformed( null );
                }
          if( this.compHashtable.containsKey( selectedCompID ) ){
            this.setCompInfo( (ComponentType)this.compHashtable.get(
selectedCompID ) );
          }
        }
      }

    /**
      * List all existing EHL objects in the project
      * and allow a user to view or edit them
      */
```

```
        void
existedEHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
    if( event.getStateChange() == ItemEvent.SELECTED ){
      String selectedEHLName = (String)event.getItem();
      this.selectedEHLIndex =
this.existedEHLComboBox.getSelectedIndex();
      if( this.selectedEHLIndex == 0 ){
        //Clean up the frame
            this.EHLClearButton_actionPerformed( null );
          }
      if( this.EHLHashtable.containsKey( selectedEHLName ) ){
        this.setEHLInfo( (EHL)this.EHLHashtable.get(
selectedEHLName ) );
      }
    }
        }
  /**
   * List all existing step type objects in a selected project
   * and allow a user to view, edit, or set secondary input
   */
  void depenStepComboBox_itemStateChanged(java.awt.event.ItemEvent
event)
        {
    Dependency dep = null;
    String selectedDepStep = null;
    if( event != null ){
      selectedDepStep = ((String)event.getItem()).trim();
    }
    this.testIndex++;
    String loopName =
(String)this.depenEHLComboBox.getSelectedItem();
    String stepName =
(String)this.depenStepComboBox.getItemAt(this.selectedDepenStepIndex);
    String output = (String)this.depenOutputTextField.getText();
    String primary = (String)this.depenPrimaryTextField.getText();
    String secondary = (String)this.depenSecondaryTextField.getText();
    if( this.testIndex < 2 ){
      this.selectedDepenStepIndex =
this.depenStepComboBox.getSelectedIndex();
      if( (secondary != null) && (this.selectedDepenStepIndex > 0) ){
        if( this.depenHashtable.containsKey( stepName ) ){
          dep = (Dependency) this.depenHashtable.get( stepName );
          if( !secondary.equals("") ){
            dep.setSecondaryInput( secondary );
          }
        }
        else{
          dep = new Dependency( loopName, stepName, output,
primary, secondary);
          this.depenHashtable.put( stepName, dep );
        }
        this.depenSecondaryTextField.setText("");
      }
    }
    else {
      if( (stepName != null) &&
(this.depenHashtable.containsKey(stepName )) ){
        dep = (Dependency) this.depenHashtable.get( stepName );
        if( !secondary.equals("") ){
          dep.setSecondaryInput( secondary );
        }
      }
      this.testIndex = 0;
    }
    if( selectedDepStep.equals(STEP_TYPE_TITLE) ){
      this.depenOutputTextField.setText("");
      this.depenPrimaryTextField.setText("");
      this.depenSecondaryTextField.setText("");
    }
    else{
      this.setDepInfo( selectedDepStep );
    }
```

```
        }
    /**
     * List all existing EHL name in the project
     * and allow a user to see its all step type objects
     */
        void
depenEHLComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == ItemEvent.SELECTED ){
       String selectedItem = (String)event.getItem();
       int selectedIndex = this.depenEHLComboBox.getSelectedIndex();

       if( this.EHLHashtable.containsKey( selectedItem ) ){
         EHL ehl = (EHL)this.EHLHashtable.get( selectedItem );

         StringTokenizer st = new StringTokenizer(
(String)ehl.getEHLPath(), "," );
                Vector tokenizeVector = new Vector();
                while( st.hasMoreTokens() ){
                  tokenizeVector.addElement( st.nextToken() );
                }
         this.setDependStepComboBox( tokenizeVector );
       }
     }
        }
    /**
     * Set step type object list in EHL panel
     *
     * @param stepVector : all existing step type objects
     */
        public void setListModel( Vector stepVector ){
          this.listModel.removeAllElements();
          for( int i=0; i< stepVector.size(); i++ ){
            StepType st = (StepType) stepVector.elementAt( i );
            this.listModel.addElement( st.getStepID() );
          }
        }
```



```
        class SymMouse extends java.awt.event.MouseAdapter
        {
                public void mouseClicked(java.awt.event.MouseEvent
event)
                {
                        Object object = event.getSource();
                        if (object == stepTypesList)
                                stepTypesList_mouseClicked(event);
                }
}
  /**
    * Monitor mouse click action on stepTypesList
    */
        void stepTypesList_mouseClicked(java.awt.event.MouseEvent
event)
        {
          String s =
((String)this.stepTypesList.getSelectedValue()).trim();
          String EHLStep = (String) this.EHLPathTextField.getText();
          Vector v = new Vector();
          if( !EHLStep.equals("") ){
       StringTokenizer st = new StringTokenizer(EHLStep, ",");
       while( st.hasMoreTokens() ){
         String s1 = ((String) st.nextToken()).trim();
         v.addElement(s1);
       }
       if( !v.contains(s) ){
                this.EHLPathTextField.setText(EHLStep+", "+s);
                v.addElement(s);
       }
            else{
         JOptionPane.showMessageDialog(this, s+" is already in this
process!",
                        "Error", JOptionPane.ERROR_MESSAGE);
          }
        }
```

```
            else{
               this.EHLPathTextField.setText(s);
            }
}
  /**
    * Save the last selected dependency object with
    * its content before quit the dependency panel
    */
         void saveLastDep(){
     Dependency dep = null;
     String loopName =
(String)this.depenEHLComboBox.getSelectedItem();
     String stepName =
(String)this.depenStepComboBox.getItemAt(this.selectedDepenStepIndex);
     String output = (String)this.depenOutputTextField.getText();
     String primary = (String)this.depenPrimaryTextField.getText();
     String secondary = (String)this.depenSecondaryTextField.getText();

     this.selectedDepenStepIndex =
this.depenStepComboBox.getSelectedIndex();
     if( (secondary != null) && (this.selectedDepenStepIndex > 0) ){
       if( this.depenHashtable.containsKey( stepName ) ){
          dep = (Dependency) this.depenHashtable.get( stepName );
          if( !secondary.equals("") ){
             dep.setSecondaryInput( secondary );
          }
       }
       else{
          dep = new Dependency( loopName, stepName, output, primary,
secondary);
          this.depenHashtable.put( stepName, dep );
       }
     }
}
  /**
    * Launch ListDialog where lists all existing component type names
    */
         void
secondaryButton_actionPerformed(java.awt.event.ActionEvent event)
         {
     (new ListDialog(this, "Component Types List",
this.compVector)).setVisible(true);
}
  /**
    * Set all selected component type names from ListDialog
    * into secondary input text field in dependency panel
    *
    * @param objs : current selected items from ListDialog
    */
         public void setItemList(Object[] objs){
           if( objs.length >= 1 ){
             String s = (String)objs[0];
             for( int i=1; i<objs.length; i++ ){
               s = s + ", "+ objs[i];
             }
             this.depenSecondaryTextField.setText(s);
           }
     }
}
```

```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * ReviewComponentContentDialog : View all links in a component
content of selectted step
 *
 * Implement CasesTitle where stores all global variables of Cases package
 */
////////////////////////////////////////////////////////////////////////
public class ReviewComponentContentDialog extends
com.sun.java.swing.JDialog implements CasesTitle
{
  /**
   * listModel : list all links of a selected link type
   */
  DefaultListModel listModel = new DefaultListModel();

  /**
   * storedVector : contains links of all available link types in the
   * component content of selected step
   */
  public Vector[] storedVector = {new Vector(), new Vector(), new
Vector(), new Vector(), new Vector(), new Vector()};

      /**
   * Build ReviewComponentContentDialog
   */
      public ReviewComponentContentDialog(Frame parent)
      {
            super(parent);
            // This code is automatically generated by Visual Cafe
when you add
            // components to the visual environment. It instantiates
and initializes
            // the components. To modify the code, only use code
syntax that matches
            // what Visual Cafe can generate, or Visual Cafe may be
unable to back
            // parse your Java file into its visual environment.
            //{{INIT_CONTROLS
            setTitle("Review Component Content");
            setModal(true);
            getContentPane().setLayout(null);
            setSize(500,450);
            setVisible(false);

      JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
            JLabel1.setText("Available Links");
            getContentPane().add(JLabel1);
            JLabel1.setForeground(java.awt.Color.black);
            JLabel1.setBounds(15,162,470,24);
            getContentPane().add(connectionComboBox);
            connectionComboBox.setBounds(115,60,270,24);
            itemScrollPane.setOpaque(true);
            getContentPane().add(itemScrollPane);
            itemScrollPane.setBounds(15,190,470,200);
            itemScrollPane.getViewport().add(itemList);
            itemList.setBounds(0,0,467,197);
            exitButton.setText("Exit");
            exitButton.setActionCommand("Cancel");
            getContentPane().add(exitButton);
            exitButton.setBounds(212,400,75,24);
            connectButton.setText("Connect");
            connectButton.setActionCommand("jbutton");
            getContentPane().add(connectButton);
            connectButton.setBounds(400,120,85,24);
```

```
        titleLabel.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.CENTER);
                titleLabel.setText("jlabel");
                getContentPane().add(titleLabel);
                titleLabel.setForeground(java.awt.Color.black);
                titleLabel.setFont(new Font("Dialog", Font.BOLD, 18));
                titleLabel.setBounds(25,10,450,30);
                selectedTextField.setEditable(false);
                getContentPane().add(selectedTextField);
                selectedTextField.setBackground(java.awt.Color.white);
                selectedTextField.setBounds(16,120,384,24);
                //}}
                        //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymItem lSymItem = new SymItem();
                connectionComboBox.addItemListener(lSymItem);
                SymAction lSymAction = new SymAction();
                connectButton.addActionListener(lSymAction);
                exitButton.addActionListener(lSymAction);
                SymMouse aSymMouse = new SymMouse();
                itemList.addMouseListener(aSymMouse);
                //}}

          itemList.setModel(listModel);
    connectionComboBox.addItem(LINKS_TITLE);
          for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
      connectionComboBox.addItem(LINK_FILE_NAMES[i]);
    }
  }

        public ReviewComponentContentDialog()
        {
                this((Frame)null);
        }

        /**
    * After select a step and press OK button from ListDialog, it will be
launched
    */
   public ReviewComponentContentDialog(String title, Vector[]
storedVector){
      this();
      this.titleLabel.setText(title);
      this.storedVector = storedVector;
   }

   public ReviewComponentContentDialog(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new
ReviewComponentContentDialog()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();
```

```
                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets
                Insets insets = getInsets();
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox connectionComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JScrollPane itemScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JList itemList = new
com.sun.java.swing.JList();
        com.sun.java.swing.JButton exitButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton connectButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel titleLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField selectedTextField = new
com.sun.java.swing.JTextField();
        //}}

        class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == connectionComboBox)

        connectionComboBox_itemStateChanged(event);
                }
        }
  /**
   * Select a link type
   */
        void
connectionComboBox_itemStateChanged(java.awt.event.ItemEvent event)
        {
          if( event.getStateChange() == event.SELECTED ){
          selectedTextField.setText("");
          listModel.removeAllElements();
            String selectedItem = (String)event.getItem();
            for( int i=0; i<LINK_FILE_NAMES.length; i++ ){
         if( selectedItem.equals(LINK_FILE_NAMES[i]) ){
           for( int j=0; j<storedVector[i].size(); j++ ){
             listModel.addElement(storedVector[i].elementAt(j));
           }
           i= LINK_FILE_NAMES.length;
         }
       }
        }
        }

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == connectButton)
```

```
                    connectButton_actionPerformed(event);
                else if (object == exitButton)
                    exitButton_actionPerformed(event);
            }
        }
    /**
     * Connect to an application which matchs with selected item from
connectionComboBox
     * and view the file which the name is the content of selectedTextField
link
     */
        void connectButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      String s = " " + selectedTextField.getText();
      if( !s.equals("") ){
        if( connectionComboBox.getSelectedIndex()>0 ){
          int index = connectionComboBox.getSelectedIndex()-1;
          if( index == 3 ){
              (new PersonnelFrame(s.trim(), "View")).setVisible(true);
          }
          else{
            try {
              Runtime runtime = Runtime.getRuntime();
              runtime.exec(EXECUTIONS[index]+s);
            }
            catch( Exception e ) {
              throw new RuntimeException(e.toString());
            }
          }
        }
      }
        }
    /**
     * Exit ReviewComponentContentDialog
     */
        void exitButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
      setVisible( false );
      dispose();
        }

        class SymMouse extends java.awt.event.MouseAdapter
        {
            public void mouseClicked(java.awt.event.MouseEvent
event)
            {
                Object object = event.getSource();
                if (object == itemList)
                    itemList_mouseClicked(event);
            }
        }
    /**
     * Set the selectedTextField when a user selects an item from this list
     */
        void itemList_mouseClicked(java.awt.event.MouseEvent event)
        {
          if( event.getClickCount() > 0) {
            String s = (String)this.itemList.getSelectedValue();
            this.selectedTextField.setText(s);
          }
    }
}
```

```
package Cases;

import java.awt.*;
import com.sun.java.swing.*;
import com.sun.java.swing.table.*;
import java.awt.event.*;
import java.util.*;

////////////////////////////////////////////////////////////////////////
/**
 * SkillTableFrame : a table of skill IDs, skill names, and skill levels
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_Cases
 */
////////////////////////////////////////////////////////////////////////
public class SkillTableFrame extends com.sun.java.swing.JFrame
implements CasesTitle
{
  /**
   * scf : parent frame to launch this frame
   */
  StepContentFrame scf = null;

  /**
   * pf : parent frame to launch this frame
   */
  PersonnelFrame pf = null;

  /**
   * skillVector : selected skill to return to parent frame
   */
  private Vector skillVector = new Vector();

  /**
   * skillModel : monitor skill table
   */
  protected DefaultTableModel skillModel = null;

  /**
   * Build SkillTableFrame
   */
        public SkillTableFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Skill List");
                getContentPane().setLayout(null);
                setSize(405,305);
                setVisible(false);
                tableScrollPane.setOpaque(true);
                getContentPane().add(tableScrollPane);
                tableScrollPane.setBounds(33,54,324,169);
                tableScrollPane.getViewport().add(skillTable);
                skillTable.setBounds(0,0,321,166);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel1.setText("Skill List");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 16));
                JLabel1.setBounds(15,6,375,47);
                OKButton.setText("OK");
                OKButton.setActionCommand("OK");
                getContentPane().add(OKButton);
```

```
                OKButton.setBounds(121,252,73,24);
                cancelButton.setText("Cancel");
                cancelButton.setActionCommand("Cancel");
                getContentPane().add(cancelButton);
                cancelButton.setBounds(211,252,73,24);
                //}}

                //{{INIT_MENUS
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                OKButton.addActionListener(lSymAction);
                cancelButton.addActionListener(lSymAction);
                //}}

  /**
    * create the new skill table
    */
                createSkillTable();
        }

/**
 * StepContentFrame launch this frame when skill button is slected
 */
public SkillTableFrame(StepContentFrame scf){
   this();
   this.scf = scf;
}
/**
 * PersonnelFrame launch this frame  when skill button is slected
 */

public SkillTableFrame(PersonnelFrame pf){
   this();
   this.pf = pf;
}

        public SkillTableFrame(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new SkillTableFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
```

```
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JScrollPane tableScrollPane = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JTable skillTable = new
com.sun.java.swing.JTable();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton OKButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton cancelButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == OKButton)
                                OKButton_actionPerformed(event);
                        else if (object == cancelButton)
                                cancelButton_actionPerformed(event);
                }
        }

  /**
   * Return all selected skills to a parent frame and exit SkillTableFrame
   */
        void OKButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          int[] selectedRows = skillTable.getSelectedRows();
          for( int i=0; i<selectedRows.length; i++ ){
            int index = selectedRows[i];
        String s = skillModel.getValueAt(index,0)+" :
"+skillModel.getValueAt(index,1)+" : "+skillModel.getValueAt(index,2);
            skillVector.addElement(s);
          }
          if( this.scf != null ){
            this.scf.setSkillComboBox(skillVector);
          }
          else if( this.pf != null ){
            this.pf.setSkillComboBox(skillVector);
          }
          setVisible(false);
          dispose();
        }

  /**
   * Exit this frame without return anything
   */
        void cancelButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
          setVisible(false);
          dispose();
        }

  /**
```

```
    * Create a skill table with 3 columns and 21 rows
    */
        void createSkillTable(){
                String[] skillColumnNames = {"Skill ID", "Skill Name",
"Skill Level" };
                final Object[][] skillData = new Object[20][3];

                for( int i = 1; i <SKILL_ID; ++i ){
                        skillData[i-1][0] = ""+i;
                }

                for( int j=0; j<SKILL_LIST.length; j++ ){
                   skillData[j][1] = SKILL_LIST[j];
                   skillData[j][2]=""+0;
                }
            skillModel = new DefaultTableModel(skillData,
skillColumnNames);
                JComboBox skillLevelComboBox = new JComboBox();
                for( int k= 0; k <SKILL_LEVEL; ++k ){
                        skillLevelComboBox.addItem(""+k);
                }
            skillTable.setModel(skillModel);

                TableColumn skillLevelColumn =
skillTable.getColumn(skillColumnNames[2]);

                // Use the combo box as the editor in the "Skill Level"
column.
                skillLevelColumn.setCellEditor(new
DefaultCellEditor(skillLevelComboBox));
        }
}
```



```
package Cases;

/**
 * Step Content Object which is used to save in step.cnt file
 */

import java.io.Serializable;
import java.util.*;

//////////////////////////////////////////////////////////////////////
/**
 * StepContent : Create a step content object and save it in step.cnt file
 */
//////////////////////////////////////////////////////////////////////
public class StepContent implements Serializable{

  /**
   * stepName : step version
   */
  private String stepName = null;

  /**
   * status : status of the step, eg. approved, scheduled,complete,...
   */
  private String status = null;

  /**
   * skill : a vector of skills
   */
  private Vector skill = new Vector();

  /**
   * securityLevel : security level of the step
   */
  private int securityLevel = 0;

  /**
```

```
 * organizer : a person's ID to organize the step
 */
private String organizer = null;

/**
 * predecessors : a vector of atomics
 */
private Vector predecessors = new Vector();

/**
 * priority : priority of the step
 */
private int priority = 0;

/**
 * estimateDuration : estimate how long the job will finish
 */
private int estimatedDuration = 0;

/**
 * deadline : deadline of the job
 */
private String deadline = null;

/**
 * earliestStartTime : the time to start in the plan
 */
private String earliestStartTime = null;

/**
 * finishTime : when the job is finished
 */
private String finishTime = null;

/**
 * realStartTime : the actual day to start the job
 */
private String realStartTime = null;

/**
 * manager : a person's ID to manage the job
 */
private String manager = null;

/**
 * evaluation : description of an evaluation
 */
private String evaluation = null;

/**
 * evaluator : a person's ID to evaluate the job
 */
private String evaluator = null;

/**
 * This StepContent constructor is used to create a step content object
 */
public StepContent( String stepName, String status, Vector skill,
                    int securityLevel, String evaluation, String evaluator,
                    String organizer,Vector predecessors, int priority,
                    int estimatedDuration, String deadline, String startTime,
                    String finishTime,  String manager){
  this.stepName = stepName;
  this.status = status;
  this.skill = skill;
  this.securityLevel = securityLevel;
  this.evaluation = evaluation;
  this.evaluator = evaluator;
  this.organizer = organizer;
  this.predecessors = predecessors;
  this.priority = priority;
  this.estimatedDuration = estimatedDuration;
  this.deadline = deadline;
  this.earliestStartTime = startTime;
```

```
    this.finishTime = finishTime;
    this.manager = manager;
}

/**
 * Empty StepContent constructor
 */
public StepContent(){
}

/**
 * Set a name for the step
 *
 * @param s : name of the step
 */
public void setStepName(String s){
    this.stepName = s;
}

/**
 * Get the step's name
 *
 * @return stepName : the step's name
 */
public String getStepName(){
    return this.stepName;
}

/**
 * Set status for the step
 *
 * @param s : status of the step
 */
public void setStatus(String s){
    this.status = s;
}

/**
 * Status of the step
 *
 * @return status : status of the step
 */
public String getStatus(){
    return this.status;
}

/**
 * Set skills for the step
 *
 * @param v : vector of skills
 */
public void setSkill(Vector v){
    this.skill = v;
}

/**
 * Skills of the step
 *
 * @return skill : vector of skills of the step
 */
public Vector getSkill(){
    return skill;
}

/**
 * Set security level for the step
 *
 * @param i: security level
 */
public void setSecurityLevel(int i){
    this.securityLevel = i;
}

/**
```

```
 * Security level of the step
 *
 * @return securityLevel : security level of the step
 */
public int getSecurityLevel(){
   return this.securityLevel;
}

/**
 * Set evaluation of the step
 *
 * @param s : a string of evaluation
 */
public void setEvaluation(String s){
   this.evaluation = s;
}

/**
 * Evaluation of the step
 *
 * @return evaluation : evaluation of the step
 */
public String getEvaluation(){
   return this.evaluation;
}

/**
 * Set evaluator for the step
 *
 * @param s : evaluator's name
 */
public void setEvaluator(String s){
   this.evaluator = s;
}

/**
 * Evaluator of the step
 *
 * @return evaluator : evaluator of the step
 */
public String getEvaluator(){
   return this.evaluator;
}

/**
 * Set organizer for the step
 *
 * @param s : the name of organizer
 */
public void setOrganizer(String s){
   this.organizer = s;
}

/**
 * Organizer of the step
 *
 * @return organizer : organizer of the step
 */
public String getOrganizer(){
   return this.organizer;
}

/**
 * Set predecessors for the step
 *
 * @param v : vector of atomics
 */
public void setPredecessors(Vector v){
   this.predecessors = v;
}

/**
 * Predecessors of the step
 *
```

```
 * @return predecessors : a vector of atomics
 */
public Vector getPredecessors(){
   return this.predecessors;
}

/**
 * Set priority for the step
 *
 * @param i : level of priority
 */
public void setPriority(int i){
   this.priority = i;
}

/**
 * Priority of the step
 *
 * @return priority : an integer of priority level
 */
public int getPriority(){
   return this.priority;
}

/**
 * Set duration for the step
 *
 * @param i : prediction of how long it will take to finish the job
 */
public void setDuration(int i){
   this.estimatedDuration = i;
}

/**
 * Estimate duration to finish the job
 *
 * @return estimatedDuration : time to finish the job
 */
public int getDuration(){
   return this.estimatedDuration;
}

/**
 * Set deadline of the job
 *
 * @param d : exactly day to be done this job
 */
public void setDeadline(String d){
   this.deadline = d;
}

/**
 * Deadline of the job
 *
 * @return deadline : a string to express the deadline
 */
public String getDeadline(){
   return this.deadline;
}

/**
 * Set earliest start time for the job
 *
 * @param d : a string to express the earliest start time
 */
public void setStartTime(String d){
   this.earliestStartTime = d;
}

/**
 * The day to plan starting the job
 *
 * @return earliestStartTime :  plan to start on this day
 */
```

```
public String getStartTime(){
   return this.earliestStartTime;
}

/**
 * Set the finish time for the job
 *
 * @param d : a string to express the finish day
 */
public void setFinishTime(String d){
   this.finishTime = d;
}

/**
 * The finish day
 *
 * @return finishTime : a string to express the finish day
 */
public String getFinishTime(){
   return this.finishTime;
}

/**
 * Set the exactly day to start the job
 *
 * @param realStartTime : a string to express the day to start the job
 */
public void setRealStartTime(String d){
   this.realStartTime = d;
}

/**
 * The day to start the job
 *
 * @return realStartTime : a string to express the day to start the job
 */
public String getRealStartTime(){
   return this.realStartTime;
}

/**
 * Set manager's ID for the step
 *
 * @param s : the manager's ID
 */
public void setManager(String s){
   this.manager = s;
}

/**
 * The manager's ID to manage the job
 *
 * @return manager : the manager's ID
 */
public String getManager(){
   return this.manager;
}
}
```

```
package Cases;

import java.awt.*;
import java.awt.event.*;
import java.text.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;
import com.symantec.itools.awt.MaskedTextField;

////////////////////////////////////////////////////////////////////////////////
/**
 * StepContentFrame : Use to create/delete/view step content of the
selected step
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_StepContent
 */
////////////////////////////////////////////////////////////////////////////////
public class StepContentFrame extends com.sun.java.swing.JFrame
implements CasesTitle, I_StepContent
{
  /**
   * atomicsVector : contains all atomics of the selected step
   */
  public Vector atomicsVector = new Vector();

  /**
   * selectedSkill : contains all selected skills from SkillTableFrame
   */
  public Vector selectedSkill = null;

  /**
   * selectedPred : contains all selected predecessors from ListDialog
   */
  public Vector selectedPred = null;

  /**
   * pathName : the complete path of current step
   */
  public String pathName = null;

  /**
   * Build StepContentFrame
   */
        public StepContentFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("SPIDER-Step Content");
                getContentPane().setLayout(null);
                setSize(700,550);
                setVisible(false);
                getContentPane().add(saveButton);
                saveButton.setBounds(0,0,0,0);
                getContentPane().add(deleteButton);
                deleteButton.setBounds(0,0,0,0);
                exitButton.setText("Exit");
                exitButton.setActionCommand("Save");
                getContentPane().add(exitButton);
                exitButton.setBounds(590,470,75,24);

        titleLable.setHorizontalAlignment(com.sun.java.swing.SwingCons
tants.RIGHT);
                titleLable.setText("Step Content:");
                getContentPane().add(titleLable);
```

```
                titleLable.setForeground(java.awt.Color.black);
                titleLable.setFont(new Font("Dialog", Font.BOLD, 18));
                titleLable.setBounds(82,30,130,30);  //y=6
                /**stepVersionTF**/
                stepVersionTextField.setEditable(false);
                getContentPane().add(stepVersionTextField);

        stepVersionTextField.setBackground(java.awt.Color.white);

        stepVersionTextField.setBounds(102,105,200,24);//246,55,300,24)
;
                /**/
                /**stepVersionLabel**/

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel1.setText("Step Version");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(1,105,100,24);//154,55,90,24);
                /**/
                getContentPane().add(managerComboBox);
                managerComboBox.setBounds(472,405,200,24);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel2.setText("Manager");
                getContentPane().add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2.setBounds(371,405,100,24);
                /**Status label**/

        JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel3.setText("Status");
                getContentPane().add(JLabel3);
                JLabel3.setForeground(java.awt.Color.black);
                JLabel3.setBounds(1,155,100,24);//1,105,100,24);
                getContentPane().add(statusComboBox);

        statusComboBox.setBounds(102,155,200,24);//102,105,200,24);

        JLabel5.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel5.setText("Evaluation");
                getContentPane().add(JLabel5);
                JLabel5.setForeground(java.awt.Color.black);
                JLabel5.setBounds(1,305,100,24);

        JLabel6.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel6.setText("Evaluator");
                getContentPane().add(JLabel6);
                JLabel6.setForeground(java.awt.Color.black);
                JLabel6.setBounds(1,355,100,24);
                getContentPane().add(evaluatorComboBox);
                evaluatorComboBox.setBounds(102,355,200,24);

        JLabel7.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel7.setText("Security Level");
                getContentPane().add(JLabel7);
                JLabel7.setForeground(java.awt.Color.black);
                JLabel7.setBounds(1,255,100,24);
                getContentPane().add(securityLevelComboBox);
                securityLevelComboBox.setBounds(102,255,200,24);

        JLabel9.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel9.setText("Organizer");
                getContentPane().add(JLabel9);
                JLabel9.setForeground(java.awt.Color.black);
                JLabel9.setBounds(1,405,100,24);
                getContentPane().add(organizerComboBox);
```

```
                organizerComboBox.setBounds(101,405,200,24);
                getContentPane().add(predecessorsComboBox);
                predecessorsComboBox.setBounds(472,105,200,24);

        JLabel12.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel12.setText("Priority");
                getContentPane().add(JLabel12);
                JLabel12.setForeground(java.awt.Color.black);
                JLabel12.setBounds(371,155,100,24);
                getContentPane().add(priorityComboBox);
                priorityComboBox.setBounds(472,155,200,24);
                //**SkillComboBox
                getContentPane().add(skillComboBox);

        skillComboBox.setBounds(102,205,200,24);//102,155,200,24);
                //**/

        JLabel16.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.RIGHT);
                JLabel16.setText("Estimated Duration");
                getContentPane().add(JLabel16);
                JLabel16.setForeground(java.awt.Color.black);
                JLabel16.setBounds(361,205,110,24);
                getContentPane().add(evaluationTextField);
                evaluationTextField.setBounds(101,305,200,24);
                deadlineTextField.setEditable(false);
                getContentPane().add(deadlineTextField);
                deadlineTextField.setBackground(java.awt.Color.white);
                deadlineTextField.setBounds(472,255,200,24);
                earliestSTTextField.setEditable(false);
                getContentPane().add(earliestSTTextField);

        earliestSTTextField.setBackground(java.awt.Color.white);
                earliestSTTextField.setBounds(472,305,200,24);
                finishTimeTextField.setEditable(false);
                getContentPane().add(finishTimeTextField);
        finishTimeTextField.setBackground(java.awt.Color.white);
                finishTimeTextField.setBounds(472,355,200,24);
                selectedLabel.setText("jlabel");
                getContentPane().add(selectedLabel);
                selectedLabel.setForeground(java.awt.Color.black);
                selectedLabel.setFont(new Font("Dialog", Font.BOLD,
14));
                selectedLabel.setBounds(217,30,400,30);  //y=6
                predecessorsButton.setText("Predecessors");
                predecessorsButton.setActionCommand("jbutton");
                getContentPane().add(predecessorsButton);
                predecessorsButton.setBounds(351,105,120,24);
                //**SkillButton
                skillButton.setText("Skill");
                skillButton.setActionCommand("Skill");
                getContentPane().add(skillButton);
                skillButton.setBounds(21,205,80,24);//21,155,80,24);
                //**/
                deadlineButton.setText("Deadline");
                deadlineButton.setActionCommand("Deadline");
                getContentPane().add(deadlineButton);
                deadlineButton.setBounds(371,255,100,24);
                startTimeButton.setText("Earliest Start Time");
                startTimeButton.setActionCommand("Earliest Start
Time");
                getContentPane().add(startTimeButton);
                startTimeButton.setBounds(332,305,139,24);
                finishTimeButton.setText("Finish Time");
                finishTimeButton.setActionCommand("Finish Time");
                getContentPane().add(finishTimeButton);
                finishTimeButton.setBounds(371,355,100,24);
                getContentPane().add(estDurationTextField);
                estDurationTextField.setBounds(472,205,200,24);
                //}}

                //{{INIT_MENUS
```

```
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
                saveButton.addActionListener(lSymAction);
                deleteButton.addActionListener(lSymAction);
                exitButton.addActionListener(lSymAction);
                predecessorsButton.addActionListener(lSymAction);
                skillButton.addActionListener(lSymAction);
                deadlineButton.addActionListener(lSymAction);
                startTimeButton.addActionListener(lSymAction);
                finishTimeButton.addActionListener(lSymAction);
                //}}

        //Set status combobox
        statusComboBox.addItem("Proposed");
        statusComboBox.addItem("Approved");
        statusComboBox.addItem("Scheduled");
        statusComboBox.addItem("Assigned");
        statusComboBox.addItem("Decomposed");
        statusComboBox.addItem("Abandoned");
        statusComboBox.addItem("Completed");

        //Set security level from 0..5
        for( int i=0; i<SECURITY_LEVEL; i++){
           securityLevelComboBox.addItem(i+"");
        }

        //Set priority combobox from 0..5
        for( int j=0; j<PRIORITY_LEVEL; j++){
           priorityComboBox.addItem(j+"");
        }
            }

    /**
     * CasesFrame launch this frame when Step Content menu item is
selected or
     * Step Content button from Trace Frame is pressed
     *
     * @param traceFrame :  = null if launched from Step Content menu
item
     *                  != null if launched from Step Content button
     * @param pathName : the path of current step
     * @param atomics : vector of the current step's atomics
     */
    public StepContentFrame(TraceFrame traceFrame, String stepName,
String pathName, Vector atomics)
            {
        this();
                    this.atomicsVector = atomics;
                    this.selectedLabel.setText( pathName );
                    this.stepVersionTextField.setText( stepName );
                    this.pathName = pathName;
                    setStakeHolders();
                symantec.itools.awt.util.Calendar theCalendar = new
symantec.itools.awt.util.Calendar();
                    try{
                       File f = new File(pathName+"\\step.cnt");
                       if( f.exists() ){
                          FileInputStream fileInput = new FileInputStream(f);
                          ObjectInputStream oi = new ObjectInputStream(
fileInput );
                          if( oi != null ){
                             setInitial((StepContent)oi.readObject());
                          }
                          else{
                             try{
                             DateFormat df = DateFormat.getDateInstance();
                             Date d = df.parse(theCalendar.getDate());
                deadlineTextField.setText(d.toString());
                earliestSTTextField.setText(d.toString());
                finishTimeTextField.setText(d.toString());
                }
                catch(Exception e){System.out.println(e);}
```

```
                }
                oi.close();
                fileInput.close();
}
            }
            catch( IOException io ){
              debug("IOException: "+io);
              try{
                DateFormat df = DateFormat.getDateInstance();
                Date d = df.parse(theCalendar.getDate());
  deadlineTextField.setText(d.toString());
  earliestSTTextField.setText(d.toString());
  finishTimeTextField.setText(d.toString());
}
catch( Exception e){System.out.println(e);}

            }
            catch( ClassNotFoundException c ){
              debug("ClassNotFoundException: "+c);
            }
            if( traceFrame != null ){
              setReadOnly();
            }
            else{
              saveButton.setText("Save");
              saveButton.setActionCommand("Save");
              getContentPane().add(saveButton);
              saveButton.setBounds(440,470,75,24);
              deleteButton.setText("Delete");
              deleteButton.setActionCommand("Save");
              getContentPane().add(deleteButton);
              deleteButton.setBounds(515,470,75,24);
            }
  }

  public StepContentFrame(String sTitle)
  {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new StepContentFrame()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
```

```
            setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton saveButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton deleteButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton exitButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel titleLable = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField stepVersionTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox managerComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox statusComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel4 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel5 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel6 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox evaluatorComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel7 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox securityLevelComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox organizerComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox predecessorsComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel12 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JComboBox priorityComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JComboBox skillComboBox = new
com.sun.java.swing.JComboBox();
        com.sun.java.swing.JLabel JLabel16 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField evaluationTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField deadlineTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField earliestSTTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField finishTimeTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel selectedLabel = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton predecessorsButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton skillButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton deadlineButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton startTimeButton = new
com.sun.java.swing.JButton();
```

```
        com.sun.java.swing.JButton finishTimeButton = new
com.sun.java.swing.JButton();
        //com.sun.java.swing.JTextField estDurationTextField = new
com.sun.java.swing.JTextField();
        MyTextField estDurationTextField = new MyTextField();
        //}}

        //{{DECLARE_MENUS
        //}}

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == saveButton)
                                saveButton_actionPerformed(event);
                        else if (object == deleteButton)
                                deleteButton_actionPerformed(event);
                        else if (object == exitButton)
                                exitButton_actionPerformed(event);
                        else if (object == predecessorsButton)

        predecessorsButton_actionPerformed(event);
                        else if (object == skillButton)
                                skillButton_actionPerformed(event);
                        else if (object == deadlineButton)

        deadlineButton_actionPerformed(event);
                        else if (object == startTimeButton)

        startTimeButton_actionPerformed(event);
                        else if (object == finishTimeButton)

        finishTimeButton_actionPerformed(event);
                }
```



```
        }

  /**
    * Create a step content object and save it in step.cnt file
    * under the current step path.  Then exit this frame
    */
          public void
saveButton_actionPerformed(java.awt.event.ActionEvent event)
          {
      try{
        FileOutputStream fileOutput = new
FileOutputStream(this.pathName+"\\step.cnt");
        ObjectOutputStream oo = new ObjectOutputStream(fileOutput);
        if( oo != null ){
          oo.writeObject((StepContent)getStepContent());
        }
        oo.flush();
        oo.close();
        fileOutput.close();
      }
      catch(IOException io){
        debug("IOException: "+io);
      }
      exitButton_actionPerformed(event);
          }

  /**
    * Confirm message will ask a user before agree to delete step.cnt file
    */
          public void
deleteButton_actionPerformed(java.awt.event.ActionEvent event)
          {
            File f = new File(this.pathName, "step.cnt");
            if( f.exists() ){
              int result = JOptionPane.showConfirmDialog( this, "step.cnt
file will be deleted.  Would you like to continue?",
```

```
                        "Confirm
Message",JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
        //result = 0 ==> yes
        //result = 1 ==> no
        if( result == 0 ){
           f.delete();
           exitButton_actionPerformed(event);
                }
             }
             else{
                JOptionPane.showMessageDialog(this, "step.cnt file does not
exist!",
                        "Warning
Message",JOptionPane.INFORMATION_MESSAGE);
             }
          }

  /**
   * Exit this frame
   */
        public void
exitButton_actionPerformed(java.awt.event.ActionEvent event)
        {
           setVisible( false );
           dispose();
        }

  /**
   * Set initial of this frame if the selected step already created step.cnt file
   *
   * @param stepContent : step content object of this current step
   */
        public void setInitial( StepContent stepContent ){
           Vector temp = new Vector();

this.stepVersionTextField.setText((String)stepContent.getStepName());

this.statusComboBox.setSelectedItem((String)stepContent.getStatus());

           //Set up skill comboBox
           temp = (Vector)stepContent.getSkill();
     if( temp != null ){
           this.selectedSkill = new Vector();
           this.skillComboBox.addItem(SKILL_TITLE);
           for( int i=0; i<temp.size(); i++ ){
              this.skillComboBox.addItem(temp.elementAt(i));
              this.selectedSkill.addElement(temp.elementAt(i));
           }
        }

this.securityLevelComboBox.setSelectedItem(""+stepContent.getSecurityL
evel());

this.evaluationTextField.setText((String)stepContent.getEvaluation());

this.evaluatorComboBox.setSelectedItem((String)stepContent.getEvaluator(
));

this.organizerComboBox.setSelectedItem((String)stepContent.getOrganizer
());

           //Set up predecessors comboBox
           temp = new Vector();
           temp = (Vector)stepContent.getPredecessors();
     if( temp != null ){
           this.selectedPred = new Vector();
           this.predecessorsComboBox.addItem("Current Selected List");
           for( int i=0; i<temp.size(); i++ ){
              this.predecessorsComboBox.addItem(temp.elementAt(i));
              this.selectedPred.addElement(temp.elementAt(i));
           }
```

```
            }
this.priorityComboBox.setSelectedItem(""+stepContent.getPriority());
this.estDurationTextField.setText(""+stepContent.getDuration());
this.deadlineTextField.setText((String)stepContent.getDeadline());
this.earliestSTTextField.setText((String)stepContent.getStartTime());
this.finishTimeTextField.setText((String)stepContent.getFinishTime());
this.managerComboBox.setSelectedItem((String)stepContent.getManager())
;
        }

  /**
   * Create a step content object before save it
   *
   * @return stepContent object with all the information from this frame
   */
        public StepContent getStepContent(){
           int security = (new
Integer((String)this.securityLevelComboBox.getSelectedItem())).intValue()
;
           int priority = (new
Integer((String)this.priorityComboBox.getSelectedItem())).intValue();
           String estDur = (String)this.estDurationTextField.getText();
           int duration = 0;
           if( !estDur.equals("") ){
              duration = (new Integer(estDur)).intValue();
           }
           StepContent stepContent = new StepContent();


stepContent.setStepName((String)this.stepVersionTextField.getText());
stepContent.setStatus((String)this.statusComboBox.getSelectedItem());
           stepContent.setSkill((Vector)this.selectedSkill);
           stepContent.setSecurityLevel(security);

stepContent.setEvaluation((String)this.evaluationTextField.getText());

stepContent.setEvaluator((String)this.evaluatorComboBox.getSelectedItem(
));

stepContent.setOrganizer((String)this.organizerComboBox.getSelectedItem
());
           stepContent.setPredecessors((Vector)this.selectedPred);
           stepContent.setPriority(priority);
           stepContent.setDuration(duration);

stepContent.setDeadline((String)this.deadlineTextField.getText());

stepContent.setStartTime((String)this.earliestSTTextField.getText());

stepContent.setFinishTime((String)this.finishTimeTextField.getText());

stepContent.setManager((String)this.managerComboBox.getSelectedItem())
;

           return stepContent;
        }

  /**
   * Get all personnel objects in stakeholder directory,
   * and add them in evaluator, organizer, and manager combo boxes
   */
        public void setStakeHolders(){
           String[] list = (String[])STAKEHOLDER.list();
           if( list.length > 0 ){
              for( int i=0; i<list.length; i++ ){
            this.evaluatorComboBox.addItem(list[i]);
```

```
            this.organizerComboBox.addItem(list[i]);
            this.managerComboBox.addItem(list[i]);
                }
              }
            }

    /**
      * This function is called when SkillTableFrame's OK button is pressed,
      * it will return a vector of selected skills.  Use this vector to set
      * skillComboBox
      *
      * @param v : vector of skills from SkillTableFrame
      */
    public void setSkillComboBox(Vector v){
       this.selectedSkill = new Vector();
       this.skillComboBox.removeAllItems();
       if( v.size() > 0 ){
              this.skillComboBox.addItem(SKILL_TITLE);
                 for( int i=0; i<v.size(); i++ ){
                    this.skillComboBox.addItem(v.elementAt(i));
            this.selectedSkill.addElement(v.elementAt(i));
                 }
       }
            }

    /**
      * Launch ListDialog to allows a user to select more than one atomics
      */
           public void
predecessorsButton_actionPerformed(java.awt.event.ActionEvent event)
           {
              Vector v = new Vector();
              String parentPath = CASESDIRECTORY.getAbsolutePath();
              int length = parentPath.length();
              debug("Path length = "+length);
              for( int i=0; i<this.atomicsVector.size(); i++ ){
                 File f = (File) this.atomicsVector.elementAt(i);
                 String s = f.getAbsolutePath();
                 String sub1 = s.substring(length);
                 int index = sub1.indexOf("\\");
                 String sub2 = sub1.substring(index+1);

                 debug("Sub1 = "+sub1);
                 debug("Sub2 = "+sub2);
                 StringTokenizer st = new StringTokenizer(sub2, "\\");
                 String theAtomic = null;
                 int j=0;
                 while( st.hasMoreTokens()){
                    String theString = st.nextToken();
                    if( j==0 ){
                       theAtomic = theString;
                    }
                    else if( j==1){
                       theAtomic = theAtomic + theString;
                    }
                    else if( j==2){
                       theAtomic = theAtomic + "-" + theString;
                    }
                    else if( j>2){
                       theAtomic = theAtomic + "." + theString;
                    }
                    j++;
                 }
                 v.addElement(theAtomic);
              }
        (new ListDialog(this, "Predecessor List", v)).setVisible(true);
           }

     /**
       * Launch SkillTableFrame to allows a user to select more than one
skills
       */
           public void
skillButton_actionPerformed(java.awt.event.ActionEvent event)
```

```
        {
    (new SkillTableFrame(this)).setVisible(true);
        }

  /**
    * It is only called when Step Content button in TraceFrame is pressed.
    * The purpose is to view the content of this step, and the user won't
allow
    * to create/edit/delete/save this step content at all.
    */
        public void setReadOnly(){
                managerComboBox.setEnabled( false );
                statusComboBox.setEnabled( false );
                evaluatorComboBox.setEnabled( false );
                securityLevelComboBox.setEnabled( false );
                priorityComboBox.setEnabled( false );
                organizerComboBox.setEnabled( false );
                predecessorsButton.setEnabled( false );
                skillButton.setEnabled( false );
      evaluationTextField.setEditable( false );
      estDurationTextField.setEditable( false );
      deadlineTextField.setEditable( false );
      earliestSTTextField.setEditable( false );
      finishTimeTextField.setEditable( false );

        evaluationTextField.setBackground(java.awt.Color.white);

        estDurationTextField.setBackground(java.awt.Color.white);
                deadlineTextField.setBackground(java.awt.Color.white);

        earliestSTTextField.setBackground(java.awt.Color.white);

        finishTimeTextField.setBackground(java.awt.Color.white);
          startTimeButton.setEnabled( false );
          deadlineButton.setEnabled( false );
          finishTimeButton.setEnabled( false );
        }

  /**
    * Launch CalendarDialog
    */
        public void
startTimeButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          String s = earliestSTTextField.getText();
          if( !s.equals("") ){
            try{
              (new CalendarDialog(this,s,0)).setVisible(true);
            }
            catch(Exception e){System.out.println(e);}
      }
      else{
              (new CalendarDialog(this,s,0)).setVisible(true);
      }
          }

  /**
    * Launch CalendarDialog
    */
        public void
deadlineButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          String s = deadlineTextField.getText();
          if( !s.equals("") ){
            try{
              (new CalendarDialog(this,s,1)).setVisible(true);
            }
            catch(Exception e){System.out.println(e);}
      }
      else{
              (new CalendarDialog(this,s,1)).setVisible(true);
      }
          }
```

```
  /**
   * Launch CalendarDialog
   */
        public void
finishTimeButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          String s = finishTimeTextField.getText();
          if( !s.equals("") ){
            try{
              (new CalendarDialog(this,s,2)).setVisible(true);
            }
            catch(Exception e){System.out.println(e);}
    }
    else{
            (new CalendarDialog(this,s,2)).setVisible(true);
    }
        }

  /**
   * Short cut to print the output
   * @param string : the output string
   */
        void debug(String s){
          System.out.println(s);
        }

        /**
         * Custom JTextfield.  It is used to create estDurationTextField
since
         * it only allows a user to input integer. And, JTextField doesn't
have
         * that capability.
         */
  public class MyTextField extends JTextField
  {
   protected void processKeyEvent( KeyEvent e )
   {
      if ( e.getModifiers() == 0 )
      {
        int keyChar = e.getKeyChar();
        if ( keyChar >= '0' && keyChar <= '9' ||
             keyChar == '\n' ){
          super.processKeyEvent( e );
        }
      }
    }
  }

  /**
   * Get an array of selected atomics from ListDialog and set
predecessorsComboBox
   *
   * @param oa : an array of selected atomics, and elements of this array
are File objects
   */
  public void setPredecessorComboBox(Object[] oa){
            this.predecessorsComboBox.removeAllItems();
            this.selectedPred = new Vector();
    if( oa.length > 0 ){
            this.predecessorsComboBox.addItem("Current Selected List");
      for( int i=0; i<oa.length; i++ ){
              this.predecessorsComboBox.addItem(oa[i]);
              this.selectedPred.addElement(oa[i]);
            }
    }
  }
}
```

```
package Cases;

/**
 * Step Type Object which is used to save in step.cfg file
 */

import java.io.Serializable;

//////////////////////////////////////////////////////////////////////////
/**
 * StepType : Create a step type object and save it in step.cfg file
 */
//////////////////////////////////////////////////////////////////////////
public class StepType implements Serializable{

   /**
    * stepID : step type ID
    */
   private String stepID = null;

   /**
    * stepName : name of the step type
    */
   private String stepName = null;

   /**
    * stepDescription : description of the step
    */
   private String stepDescription = null;

   /**
    * This StepType constructor is used to create step type object
    */
   public StepType( String stepID, String stepName, String stepDescription
){
      this.stepID = stepID;
      this.stepName = stepName;
      this.stepDescription = stepDescription;
   }

   public StepType() {}

   /**
    * Step type ID
    *
    * @param stepID : step type ID
    */
   public String getStepID(){
      return this.stepID;
   }

   /**
    * Step type name
    *
    * @param stepName : name of the step type
    */
   public String getStepName(){
      return this.stepName;
   }

   /**
    * Step type description
    *
    * @param stepDescription : the description of step type
    */
   public String getStepDescription(){
      return this.stepDescription;
   }
}
```

```
package Cases;

import java.awt.*;
import java.util.*;
import java.io.*;
import com.sun.java.swing.*;

////////////////////////////////////////////////////////////////////////
/**
 * TraceFrame : the main purpose of this frame is to view and trace the step
and
 * substeps
 *
 * Implement CasesTitle where stores all global variables of Cases package
 * Implements interface I_Trace
 */
////////////////////////////////////////////////////////////////////////
public class TraceFrame extends com.sun.java.swing.JFrame implements
CasesTitle, I_Trace
{
  /**
   * casesFrame : parent frame which launch this frame
   */
  CasesFrame casesFrame = null;

  /**
   * storedVector : stores contents of link files
   */
  public Vector[] storedVector = {new Vector(), new Vector(), new
Vector(), new Vector(), new Vector(), new Vector()};

  /**
   * currentIndex : monitor the position of current step in the history vector
   */
  public int currentIndex = 0;

  /**
   * currentLocation : make sure the current step is inserted at the right
position in the history vector
   */
  public int currentLocation = 0;

  /**
   * pathName : the path of the current project
   */
  public String pathName = null;

  /**
   * history : a vector of all selected steps in this frame
   */
  public Vector history = new Vector();

  /**
   * fnList : a vector of steps, strings, in the current project
   */
  public Vector fnList = new Vector();

  /**
   * currentPath : a string to keep the current step when tracing around
   */
  public String currentPath = null;

  /**
   * output : holds the output of the current step
   */
  private String output = "";

  /**
   * stepVersion : holds the selected step version
   */
  private String stepVersion = "";

  /**
   * Build TraceFrame
```

```
*/
        public TraceFrame()
        {
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("SPIDER-Trace");
                getContentPane().setLayout(null);
                setSize(670,320);
                setVisible(false);
                forwardButton.setText("Forward");
                forwardButton.setActionCommand("Forward");
                getContentPane().add(forwardButton);
                forwardButton.setBounds(68,1,81,24);
                backwardButton.setText("Backward");
                backwardButton.setActionCommand("Backward");
                getContentPane().add(backwardButton);
                backwardButton.setBounds(150,1,92,24);
                homeButton.setText("Home");
                homeButton.setActionCommand("Home");
                getContentPane().add(homeButton);
                homeButton.setBounds(1,1,67,24);
                stepVersionTextField.setEditable(false);
                getContentPane().add(stepVersionTextField);

        stepVersionTextField.setBackground(java.awt.Color.white);
                stepVersionTextField.setBounds(242,55,300,24);
                outputTextField.setEditable(false);
                getContentPane().add(outputTextField);
                outputTextField.setBackground(java.awt.Color.white);
                outputTextField.setBounds(242,100,300,24);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel1.setText("Step Version");
                getContentPane().add(JLabel1);
                JLabel1.setForeground(java.awt.Color.black);
                JLabel1.setBounds(55,55,180,24);

        JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel2.setText("Primary Input Component(s)");
                getContentPane().add(JLabel2);
                JLabel2.setForeground(java.awt.Color.black);
                JLabel2.setBounds(55,145,180,24);

        JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel3.setText("Secondary Input Component(s)");
                getContentPane().add(JLabel3);
                JLabel3.setForeground(java.awt.Color.black);
                JLabel3.setBounds(55,190,180,24);

        JLabel4.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.RIGHT);
                JLabel4.setText("Output Component");
                getContentPane().add(JLabel4);
                JLabel4.setForeground(java.awt.Color.black);
                JLabel4.setBounds(55,100,180,24);
                closeButton.setText("Close");
                closeButton.setActionCommand("OK");
                getContentPane().add(closeButton);
                closeButton.setBounds(302,252,75,24);
                stepContentButton.setText("Step Content");
                stepContentButton.setActionCommand("Step Content");
                getContentPane().add(stepContentButton);
                stepContentButton.setBounds(560,1,107,24);
```

```
        primaryTextField.setEditable(false);
        getContentPane().add(primaryTextField);
        primaryTextField.setBackground(java.awt.Color.white);
        primaryTextField.setBounds(242,145,300,24);
        secondaryTextField.setEditable(false);
        getContentPane().add(secondaryTextField);

    secondaryTextField.setBackground(java.awt.Color.white);
        secondaryTextField.setBounds(242,190,300,24);
        traceButton.setText("Trace");
        traceButton.setActionCommand("Trace");
        getContentPane().add(traceButton);
        traceButton.setBounds(242,1,67,24);
        decomposeButton.setText("Decompose");
        decomposeButton.setActionCommand("Decompose");
        getContentPane().add(decomposeButton);
        decomposeButton.setBounds(310,1,102,24);
        componentButton.setText("Component Content");
        componentButton.setActionCommand("Component
Content");
        getContentPane().add(componentButton);
        componentButton.setBounds(413,1,146,24);
        //}}

        //{{INIT_MENUS
        //}}

        //{{REGISTER_LISTENERS
        SymAction lSymAction = new SymAction();
        homeButton.addActionListener(lSymAction);
        forwardButton.addActionListener(lSymAction);
        backwardButton.addActionListener(lSymAction);
        closeButton.addActionListener(lSymAction);
        stepContentButton.addActionListener(lSymAction);
        traceButton.addActionListener(lSymAction);
        decomposeButton.addActionListener(lSymAction);
        componentButton.addActionListener(lSymAction);
        //}}

        //Grey out when the history vector is empty
        forwardButton.setEnabled( false );
        backwardButton.setEnabled( false );
    }

  /**
   * CasesFrame launch this frame when Trace menu item is selected
   */
  public TraceFrame( CasesFrame casesFrame, String pathName, Vector
componentsVector, Vector fnList ){
    this();
    this.casesFrame = casesFrame;
    this.pathName = pathName;
    this.fnList = fnList;
    setInitial(componentsVector);
  }

    public TraceFrame(String sTitle)
    {
        this();
        setTitle(sTitle);
    }

    public void setVisible(boolean b)
    {
        if (b)
            setLocation(50, 50);
        super.setVisible(b);
    }

    static public void main(String args[])
    {
        (new TraceFrame()).setVisible(true);
    }
```

```
        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton forwardButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton backwardButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton homeButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField stepVersionTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField outputTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel4 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton closeButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton stepContentButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JTextField primaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField secondaryTextField = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JButton traceButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton decomposeButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton componentButton = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}

  class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == homeButton)
                                homeButton_actionPerformed(event);
```

```
                    else if (object == closeButton)
                        closeButton_actionPerformed(event);
                    else if (object == forwardButton)
                        forwardButton_actionPerformed(event);
                    else if (object == backwardButton)

        backwardButton_actionPerformed(event);
                    else if (object == stepContentButton)

        stepContentButton_actionPerformed(event);
                    else if (object == traceButton)
                        traceButton_actionPerformed(event);
                    else if (object == decomposeButton)

        decomposeButton_actionPerformed(event);
                    else if (object == componentButton)

        componentButton_actionPerformed(event);
                }
        }

    /**
     * View the first step in the history vector
     */
        public void
homeButton_actionPerformed(java.awt.event.ActionEvent event)
        {
      this.currentIndex = 0;
      String s = (String)this.history.elementAt(this.currentIndex);
      this.currentPath = (String)convertToThePath(s);
      searchPath(s);
        }

    /**
     * View the step at the currentIndex+1 postion in the history vector
     */
        public void
forwardButton_actionPerformed(java.awt.event.ActionEvent event)
        {
      if( (++this.currentIndex >= 0 ) && (this.currentIndex <
this.history.size()) ){
        String s = (String)this.history.elementAt(this.currentIndex);
        this.currentPath = (String)convertToThePath(s);
        searchPath(s);
      }
        }

    /**
     * View the step at the currentIndex-1 postion in the history vector
     */
        public void
backwardButton_actionPerformed(java.awt.event.ActionEvent event)
        {
      if ( --this.currentIndex >= 0 ){
        String s = (String)this.history.elementAt(this.currentIndex);
        this.currentPath = (String)convertToThePath(s);
        searchPath(s);
      }
        }

    /**
     * Launch StepContentFrame to view the content of this current step
     */
        public void
stepContentButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          String s = (String)this.history.elementAt(this.currentIndex);
          this.currentPath = (String)convertToThePath(s);
      this.casesFrame.setStepContent(this, "s-"+s, this.currentPath );
    }

    /**
     * Exit TraceFrame
```

```
    */
        public void
closeButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          setVisible(false);
          dispose();
        }

  /**
   * Short cut to print the output
   * @param string : the output string
   */
        public void debug(String s){
          System.out.println(s);
        }

  /**
   * Set intitial of this frame with the selected step from JFileChooser
   *
   * @param componentsVector : a vector of strings holds all component
names of the selected step
   */
  public void setInitial(Vector componentsVector){
    String s = (String)componentsVector.elementAt(0);
    setHistory(s);
    this.currentPath = (String)convertToThePath(s);
    if( this.currentPath != null ){
      searchPath(s);
    }
  }

  /**
   * Hold all steps which have been viewed and traced
   *
   * @param s : the current step name
   */
  public void setHistory( String s){
    this.history.insertElementAt(s,currentLocation++);
  }

  /**
   * Convert from a selected step into the complete file path of this step
and
   * make sure this step is valid or not
   *
   * @param s : selected step name
   */
  public String convertToThePath(String s){
    String thePath = null;
    output = s;
    s= "s-"+s;
    stepVersion = s;

    //To convert the string of selected item into the file path
    for( int i=0; i<this.fnList.size(); i++ ){
      String fn = (String)this.fnList.elementAt(i);
      String sub = s.substring(0,fn.length());

      if( sub.equals(fn)){
        i = this.fnList.size();
        thePath= this.pathName+"\\"+sub+"\\";
        String sub2 = s.substring(fn.length());

        char[] ca = (char[])sub2.toCharArray();
        int result = 0;
        for( int j=0; j<ca.length; j++ ){
          String s3 = ca[j]+"";
          result = s3.compareTo("-");
          if( result == 0 ){
            j=ca.length;
            StringTokenizer st = new StringTokenizer(sub2,"-");
                String before_ = st.nextToken("-");
                thePath = thePath+before_;
                String _after = st.nextToken("-");
```

```
                    st = new StringTokenizer(_after,".");
                    while(st.hasMoreTokens()){
                       thePath = thePath+"\\"+st.nextToken();
                    }
                }
                else if( (result > 0) && (j==ca.length - 1) ){
                   thePath = thePath+sub2;
                }
             }
          }
       }
       if( thePath == null ){
          JOptionPane.showMessageDialog(this, s+" is invalid name.",
                      "Error Message",JOptionPane.ERROR_MESSAGE);
          return null;
       }
       return thePath;
    }

    /**
     * Search the path of this step
     *
     * @param s : the selected step name
     */
    public void searchPath( String s){
       searchIndex();
       if( this.currentPath != null ){
          searchInputFiles( this.currentPath );
       }
    }

    /**
     * Search input.p and input.s files of the current step, and get theirs
     * contents to insert in the primary and secondary textfields directly
     *
     * @param thePath : the complete path of the selected step
     */
    public void searchInputFiles( String thePath ){
       clearTextFields();
       try{
                    File f = new File(thePath+"\\input.p");
                    if( f.exists() ){
                       FileInputStream fileInputP = new
FileInputStream(f);
                       DataInputStream primIn = new DataInputStream(
fileInputP );
                       if( primIn != null ){
                          String s = (String)primIn.readLine();
                          if( (s != null) && (!s.equals("")) ){
                             this.primaryTextField.setText( s );
                          }
                       }
                       primIn.close();
                       fileInputP.close();
          }
          f = new File(thePath+"\\input.s");
          if( f.exists() ){
             FileInputStream fileInputS = new FileInputStream(f);
                       DataInputStream secondIn = new DataInputStream(
fileInputS);
                       if( secondIn != null ){
                          String s = (String)secondIn.readLine();
                          if( (s != null) && (!s.equals("")) ){
                             this.secondaryTextField.setText( s );
                          }
                       }
                       secondIn.close();
                       fileInputS.close();
          }
          this.outputTextField.setText(output);
          this.stepVersionTextField.setText(stepVersion);
                    }
                    catch( IOException io ){
          debug("IOException: "+io);
```

```
                }
    }

    /**
     * To find which button (home, forward, or backward) should be set
enabled
     * and it must match with the currentIndex in history vector
     */
    public void searchIndex(){
        if( this.currentIndex < this.history.size() ){
            if( this.currentIndex == (this.history.size()-1) ){
                this.forwardButton.setEnabled( false );
            }
            else{
                this.forwardButton.setEnabled( true );
            }
        }
        if( this.currentIndex > 0 ){
            this.backwardButton.setEnabled( true );
        }
        else{
            this.backwardButton.setEnabled( false );
        }
    }

    /**
     * Get selected item from ListDialog and DecomposeListDialog
     *
     * @param currentPath : reverse from a selected item into the path of
this component
     * @param selectedItem : a selected component names
     */
    public void setSelectedItem(String currentPath, String selectedItem){
        this.currentPath = currentPath;
        setHistory(selectedItem);
        this.currentIndex = this.history.size() - 1;
        searchPath(currentPath);
    }

    /**
     * Searching aFile and get its content to insert into one of elements of
storedVector
     *
     * @param aFile : link files
     * @param fileType : type of link file, eg. txt.link, word.link, ...
     */
        public void searchFiles(File aFile, String fileType){
      File f = new File( aFile, fileType);
      try{
        BufferedReader br = null;
        String item = null;
        if( fileType.equals(LINK_FILE_NAMES[0]) ){
          br = new BufferedReader( new FileReader(f));
          if( br != null ){
            while( (item = br.readLine()) != null ){
              this.storedVector[0].addElement(item);
            }
          }
        }
        else if( fileType.equals(LINK_FILE_NAMES[1]) ){
          br = new BufferedReader( new FileReader(f));
          if( br != null ){
            while( (item = br.readLine()) != null ){
              this.storedVector[1].addElement(item);
            }
          }
        }
        else if( fileType.equals(LINK_FILE_NAMES[2]) ){
          br = new BufferedReader( new FileReader(f));
          if( br != null ){
            while( (item = br.readLine()) != null ){
              this.storedVector[2].addElement(item);
            }
          }
```

```
    }
    else if( fileType.equals(LINK_FILE_NAMES[3]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.storedVector[3].addElement(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[4]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.storedVector[4].addElement(item);
        }
      }
    }
    else if( fileType.equals(LINK_FILE_NAMES[5]) ){
      br = new BufferedReader( new FileReader(f));
      if( br != null ){
        while( (item = br.readLine()) != null ){
          this.storedVector[5].addElement(item);
        }
      }
    }
  }
  catch( IOException io ){
    debug("IOException: "+io);
  }
    }

/**
 * Create a file with a specific string s if s is valid
 *
 * @param s : a selected string
 * @return f : a file with the name is s
 */
public File searchFilePath( String s){
  String thePath = null;

  s = "s-"+s;
  //To convert the string of selected item into the file path
  for( int i=0; i<this.fnList.size(); i++ ){
    String fn = (String)this.fnList.elementAt(i);
    String sub = s.substring(0,fn.length());

    if( sub.equals(fn)){
      i = this.fnList.size();
      thePath= sub+"\\";
      String sub2 = s.substring(fn.length());

      char[] ca = (char[])sub2.toCharArray();
      int result = 0;
      for( int j=0; j<ca.length; j++ ){
        String s3 = ca[j]+"";
        result = s3.compareTo("-");
        if( result == 0 ){
          j=ca.length;
          StringTokenizer st = new StringTokenizer(sub2,"-");
              String before_ = st.nextToken("-");
              thePath = thePath+before_;
              String _after = st.nextToken("-");
              st = new StringTokenizer(_after,".");
              while(st.hasMoreTokens()){
                thePath = thePath+"\\"+st.nextToken();
              }
        }
        else if( (result > 0) && (j==ca.length - 1) ){
          thePath = thePath+sub2;
        }
      }
    }
  }
  File f = new File(this.pathName,thePath);
```

```
    return f;
  }

  /**
   * Check a selected string is valid or invalid file name in the current
project
   *
   * @param selectedItem : a selected component name
   * @return a component name which matchs in the current project
   */
  public String checkSelection(String selectedItem ){
    if( selectedItem != null ){
      int j = selectedItem.indexOf("-");
      if( j>0){
              String sub2 = (selectedItem.substring(j+1)).trim();
              char c = (char)sub2.charAt(0);
              boolean isLetter = (new Character(c)).isLetter(c);
              if( isLetter ){
        return sub2;
        }
        else{
          return selectedItem;
        }
      }
      else{
        return selectedItem;
      }
    }
    return selectedItem;
        }

  /**
   * Tokenize a string with the delimit is ","
   *
   * @param v : a vector of words without ","
   * @param s : a string is tokenized
   */
  public void tokenizer( Vector v, String s){
          StringTokenizer st = new StringTokenizer(s,",");
          while(st.hasMoreTokens()){
            String theString = (st.nextToken()).trim();
            v.addElement(theString);
          }
        }

  /**
   * View the content of available steps in the ListDialog
   */
        public void
traceButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          Vector v = new Vector();
          tokenizer(v, primaryTextField.getText());
          tokenizer(v, secondaryTextField.getText());
          (new ListDialog(this, "Trace Component", v,
0)).setVisible(true);
        }

  /**
   * View the content of available decomposed steps of the current step
   */
        public void
decomposeButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          Vector v = new Vector();
          v.addElement(outputTextField.getText());
          tokenizer(v, primaryTextField.getText());
          tokenizer(v, secondaryTextField.getText());
          (new DecomposeListDialog(this, "Decompose",
v)).setVisible(true);
        }

  /**
   * View all links and connect these links to their applications
```

```
    * of available steps in ListDialog
    */
        public void
componentButton_actionPerformed(java.awt.event.ActionEvent event)
        {
          Vector v = new Vector();
          v.addElement(outputTextField.getText());
          tokenizer(v, primaryTextField.getText());
          tokenizer(v, secondaryTextField.getText());
          (new ListDialog(this, "Component Content", v,
1)).setVisible(true);
        }

  /**
    * Refresh all textfiels in this frame
    */
        public void clearTextFields(){
          this.stepVersionTextField.setText("");
          this.outputTextField.setText("");
          this.primaryTextField.setText("");
          this.secondaryTextField.setText("");
        }

  /**
    * Launch ReviewComponentContentDialog after select one component
from ListDialog
    *
    * @param selectedItem : a selected component name from ListDialog
    * @param f : file with the name is selectedItem
    */
        public void setComponentContent(String selectedItem, File f){
     String[] list = f.list();
     for( int j=0; j<list.length; j++ ){
       String s = (String)list[j];
       File aFile = null;
       if( s.equals(COMPONENT_CONTENT_DIR) ){
         aFile = new File(f, s);
         if( aFile.isDirectory() ){
           j = list.length;
           list = aFile.list();
           for( int k=0; k<storedVector.length; k++ ){
             storedVector[k] = new Vector();
           }
           for( int i=0; i<list.length; i++ ){
             searchFiles(aFile, (String)list[i]);
           }
           (new ReviewComponentContentDialog(selectedItem,
this.storedVector)).setVisible(true);
         }
       }
     }
        }
}


package Cases;

/**
 * Version Control Object which is used to save in the
 * current.vsn file
 */

import java.io.Serializable;

////////////////////////////////////////////////////////////////////////
/**
 * VersionControl : Create a version control object and save it in
current.vsn file
 */
////////////////////////////////////////////////////////////////////////
public class VersionControl implements Serializable{
  /**
    * currentLoop : current process of the project
```

```
    */
    private String currentLoop = null;

    /**
     * currentStep : current step of the current process
     */
    private String currentStep = null;

    /**
     * current variant : current variant of the current step
     */
    private String currentVariant = null;

    /**
     * currentVersion : current version of the current step
     */
    private String currentVersion = null;

    /**
     * currentStatus : current status of the current step, eg, Completed,
Approved, ...
     */
    private String currentStatus = "";

    /**
     * VersionControl constructor with 5 elements
     */
    public VersionControl( String currentLoop, String currentStep, String
currentVariant,
                String currentVersion, String currentStatus )
    {
      this.currentLoop = currentLoop;
      this.currentStep = currentStep;
      this.currentVariant = currentVariant;
      this.currentVersion = currentVersion;
      this.currentStatus = currentStatus;
    }

    /**
     * VersionControl constructor with 4 elements
     */
    public VersionControl( String currentLoop, String currentStep, String
currentVersion,
                String currentStatus )
    {
      this.currentLoop = currentLoop;
      this.currentStep = currentStep;
      this.currentVersion = currentVersion;
      this.currentStatus = currentStatus;
    }

    /**
     * VersionControl constructor with 3 elements
     */
    public VersionControl( String currentLoop, String currentStep, String
currentVersion )
    {
      this.currentLoop = currentLoop;
      this.currentStep = currentStep;
      this.currentVersion = currentVersion;
    }

    public VersionControl() {}

    /**
     * @return currentLoop : current process of this version control object
     */
    public String getCurrentLoop(){
      return this.currentLoop;
    }

    /**
     * @return currentStep : current step of this version control object
     */
```

```
    public String getCurrentStep(){
        return this.currentStep;
    }

    /**
     * @return currentVersion : current version of this version control object
     */
    public String getCurrentVersion(){
        return this.currentVersion;
    }

    /**
     * @return currentVariant : current variant of this version control object
     */
    public String getCurrentVariant(){
        return this.currentVariant;
    }

    /**
     * @return currentStatus : current status of this version control object
     */
    public String getCurrentStatus(){
        return this.currentStatus;
    }
}
```



```
package JobSchedule;

/*
        A basic implementation of the JDialog class.
*/

import java.awt.*;
import com.sun.java.swing.*;

public class JAboutDialog extends com.sun.java.swing.JDialog
{
    public JAboutDialog(Frame parentFrame)
    {
        super(parentFrame);
        // This code is automatically generated by Visual Cafe
when you add
        // components to the visual environment. It instantiates
and initializes
        // the components. To modify the code, only use code
syntax that matches
        // what Visual Cafe can generate, or Visual Cafe may be
unable to back
        // parse your Java file into its visual environment.
        //{{INIT_CONTROLS
        setTitle("JFC Application - About");
        setModal(true);
        getContentPane().setLayout(new GridBagLayout());
        setSize(248,94);
        setVisible(false);
        okButton.setText("OK");
        okButton.setActionCommand("OK");
        okButton.setOpaque(false);
        okButton.setMnemonic((int)'O');
        getContentPane().add(okButton, new
com.symantec.itools.awt.GridBagConstraintsD(2,1,1,1,0.0,0.0,java.awt.Gri
dBagConstraints.CENTER,java.awt.GridBagConstraints.NONE,new
Insets(0,0,10,0),0,0));
```

```
                okButton.setBounds(98,59,51,25);

        aboutLabel.setHorizontalAlignment(com.sun.java.swing.SwingCo
nstants.CENTER);
                aboutLabel.setText("A JFC Application");
                getContentPane().add(aboutLabel, new
com.symantec.itools.awt.GridBagConstraintsD(0,0,3,1,1.0,1.0,java.awt.Gri
dBagConstraints.CENTER,java.awt.GridBagConstraints.BOTH,new
Insets(0,0,0,0),0,0));
                aboutLabel.setBounds(0,0,248,59);
                //}}

                //{{REGISTER_LISTENERS
                SymWindow aSymWindow = new SymWindow();
                this.addWindowListener(aSymWindow);
                SymAction lSymAction = new SymAction();
                okButton.addActionListener(lSymAction);
                //}}
        }

        public void setVisible(boolean b)
        {
          if (b)
          {
                Rectangle bounds = (getParent()).getBounds();
                Dimension size = getSize();
                setLocation(bounds.x + (bounds.width - size.width)/2,
                                bounds.y + (bounds.height - size.height)/2);
          }

                super.setVisible(b);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension d = getSize();

                super.addNotify();

                if (fComponentsAdjusted)
                        return;
                // Adjust components according to the insets
                Insets insets = getInsets();
                setSize(insets.left + insets.right + d.width, insets.top +
insets.bottom + d.height);
                Component components[] =
getContentPane().getComponents();
                for (int i = 0; i < components.length; i++)
                {
                        Point p = components[i].getLocation();
                        p.translate(insets.left, insets.top);
                        components[i].setLocation(p);
                }
                fComponentsAdjusted = true;
        }

        // Used for addNotify check.
        boolean fComponentsAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton okButton = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel aboutLabel = new
com.sun.java.swing.JLabel();
        //}}

        class SymWindow extends java.awt.event.WindowAdapter
        {
                public void windowClosing(java.awt.event.WindowEvent
event)
                {
                        Object object = event.getSource();
```

```
                if (object == JAboutDialog.this)
                        jAboutDialog_windowClosing(event);
            }
        }

        void jAboutDialog_windowClosing(java.awt.event.WindowEvent
event)
        {
            // to do: code goes here.

            jAboutDialog_windowClosing_Interaction1(event);
        }

        void
jAboutDialog_windowClosing_Interaction1(java.awt.event.WindowEvent
event) {
            try {
                // JAboutDialog Hide the JAboutDialog
                this.setVisible(false);
            } catch (Exception e) {System.out.println(e);
            }
        }

        class SymAction implements java.awt.event.ActionListener
        {
            public void actionPerformed(java.awt.event.ActionEvent
event)
            {
                Object object = event.getSource();
                if (object == okButton)
                    okButton_actionPerformed(event);
            }
        }

        void okButton_actionPerformed(java.awt.event.ActionEvent
event)
        {
            // to do: code goes here.

            okButton_actionPerformed_Interaction1(event);
        }

        void
okButton_actionPerformed_Interaction1(java.awt.event.ActionEvent event)
{
            try {
                // JAboutDialog Hide the JAboutDialog
                this.setVisible(false);
            } catch (Exception e) {System.out.println(e);
            }
        }
}
```

```
package JobSchedule;

import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;



public class JDialog_jobskill extends com.sun.java.swing.JDialog
{
  Vector jobskill;

        public JDialog_jobskill(Frame parent)
        {
                super(parent);

                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Required Skills");
                getContentPane().setLayout(null);
                setSize(388,250);
                setVisible(false);
                JButton_delete_deleteperson.setText("Exit");

        JButton_delete_deleteperson.setActionCommand("Delete");
                getContentPane().add(JButton_delete_deleteperson);
                JButton_delete_deleteperson.setBounds(146,204,96,24);
                JLabel2.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel2.setText("Required Skills");
                getContentPane().add(JLabel2);
                JLabel2.setFont(new Font("Dialog", Font.BOLD, 15));
                JLabel2.setBounds(110,24,168,24);
                getContentPane().add(JLabel3);
                JLabel3.setBounds(74,48,240,24);
                JScrollPane1.setOpaque(true);
                getContentPane().add(JScrollPane1);
                JScrollPane1.setBounds(62,72,264,108);
                JScrollPane1.getViewport().add(JTextArea1);
                JTextArea1.setBounds(0,0,261,105);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();

        JButton_delete_deleteperson.addActionListener(lSymAction);
                //}}
        }

        public JDialog_jobskill()
        {
                this((Frame)null);
        }
        public JDialog_jobskill(Vector v)
        {
                this((Frame)null);
                jobskill=new Vector();
                jobskill=v;
                int n=jobskill.size();
                this.JTextArea1.setText("");
          this.JTextArea1.append("Skill ID"+"\t"+"Skill
Name"+"\t"+"Skill Level"+"\n");
                this.JTextArea1.append("\n");
```

```
            for(int i=0; i<jobskill.size(); i++){
               String s=(String) jobskill.elementAt(i);

               StringTokenizer st = new StringTokenizer(s,":");
        st.hasMoreTokens();
               String number=new String(st.nextToken());
               String name=st.nextToken();
               String level=st.nextToken();

        this.JTextArea1.append(number+"\t"+name+"\t"+level+"\n");

               System.out.println("job is: "+number+name+level);
            }

}


public JDialog_jobskill(String sTitle)
{
        this();
        setTitle(sTitle);
}

public void setVisible(boolean b)
{
        if (b)
                setLocation(50, 50);
        super.setVisible(b);
}

static public void main(String args[])
{
        (new JDialog_jobskill()).setVisible(true);
}

public void addNotify()
{
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets
                Insets insets = getInsets();
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton JButton_delete_deleteperson = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane JScrollPane1 = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JTextArea JTextArea1 = new
com.sun.java.swing.JTextArea();
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
```

```
                {
                        Object object = event.getSource();
                        if (object == JButton_delete_deleteperson)

        JButtonDeleteDeleteperson_actionPerformed(event);

                }
        }

        void
JButtonDeleteDeleteperson_actionPerformed(java.awt.event.ActionEvent
event)
        {
                // to do: code goes here.
                this.setVisible(false);

        }//end performence
}
```



```
package JobSchedule;

import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;


public class JDialog_message extends com.sun.java.swing.JDialog
{

   public double w;
   public Weight weight;
   JFrame_manage p;
   int n;
        public JDialog_message(Frame parent)
        {
                super(parent);
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Error");
                getContentPane().setLayout(null);
                setSize(311,122);
                setVisible(false);
                JButton_delete_deleteperson.setText("Exit");

        JButton_delete_deleteperson.setActionCommand("Delete");
                getContentPane().add(JButton_delete_deleteperson);
                JButton_delete_deleteperson.setBounds(111,84,88,24);

        JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel3.setText("No job scheduled");
                getContentPane().add(JLabel3);
                JLabel3.setBounds(72,24,167,24);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();

        JButton_delete_deleteperson.addActionListener(lSymAction);
                //}}
        }
```

```
public JDialog_message()
{
        this((Frame)null);
}


public JDialog_message(String sTitle)
{
        this();
        setTitle(sTitle);
}

public void setVisible(boolean b)
{
        if (b)
                setLocation(50, 50);
        super.setVisible(b);
}

static public void main(String args[])
{
        (new JDialog_message()).setVisible(true);
}

public void addNotify()
{
        // Record the size of the window prior to calling parents
addNotify.
        Dimension size = getSize();

        super.addNotify();

        if (frameSizeAdjusted)
                return;
        frameSizeAdjusted = true;
        // Adjust size of frame according to the insets
        Insets insets = getInsets();
        setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
}

// Used by addNotify
boolean frameSizeAdjusted = false;

//{{DECLARE_CONTROLS
com.sun.java.swing.JButton JButton_delete_deleteperson = new
com.sun.java.swing.JButton();
com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
//}}


class SymAction implements java.awt.event.ActionListener
{
        public void actionPerformed(java.awt.event.ActionEvent
event)
        {
                Object object = event.getSource();
                if (object == JButton_delete_deleteperson)

        JButtonDeleteDeleteperson_actionPerformed(event);

        }
}

void
JButtonDeleteDeleteperson_actionPerformed(java.awt.event.ActionEvent
event)
{
        this.setVisible(false);
         this.dispose();
```

```
        }
}
```

```
package JobSchedule;

import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;

public class JDialog_message1 extends com.sun.java.swing.JDialog
{

   public double w;
   public Weight weight;
   JFrame_manage p;
   int n;
        public JDialog_message1(Frame parent)
        {
                super(parent);
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Error");
                getContentPane().setLayout(null);
                setSize(311,122);
                setVisible(false);
                JButton_delete_deleteperson.setText("Exit");

        JButton_delete_deleteperson.setActionCommand("Delete");
                getContentPane().add(JButton_delete_deleteperson);
                JButton_delete_deleteperson.setBounds(111,84,88,24);

        JLabel3.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel3.setText("No job or stakeholder assigned");
                getContentPane().add(JLabel3);
                JLabel3.setBounds(36,24,239,36);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();

        JButton_delete_deleteperson.addActionListener(lSymAction);
                //}}
        }

        public JDialog_message1()
        {
                this((Frame)null);
        }


        public JDialog_message1(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
```

```
            super.setVisible(b);
        }

        static public void main(String args[])
        {
            (new JDialog_message1()).setVisible(true);
        }

        public void addNotify()
        {
            // Record the size of the window prior to calling parents
addNotify.
            Dimension size = getSize();

            super.addNotify();

            if (frameSizeAdjusted)
                return;
            frameSizeAdjusted = true;

            // Adjust size of frame according to the insets
            Insets insets = getInsets();
            setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton JButton_delete_deleteperson = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        //}}
```



```
        class SymAction implements java.awt.event.ActionListener
        {
            public void actionPerformed(java.awt.event.ActionEvent
event)
            {
                Object object = event.getSource();
                if (object == JButton_delete_deleteperson)

        JButtonDeleteDeleteperson_actionPerformed(event);

            }
        }

        void
JButtonDeleteDeleteperson_actionPerformed(java.awt.event.ActionEvent
event)
        {
            // to do: code goes here.
            this.setVisible(false);
    this.dispose();


        }
}
```

```
package JobSchedule;

import java.awt.*;
import com.sun.java.swing.*;



public class JDialog_weight extends com.sun.java.swing.JDialog
{
        public JDialog_weight(Frame parentFrame)
        {
                super(parentFrame);
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("JFC Application - About");
                setModal(true);
                getContentPane().setLayout(new GridBagLayout());
                setSize(416,162);
                setVisible(false);
                //}}

                //{{REGISTER_LISTENERS
                SymWindow aSymWindow = new SymWindow();
                this.addWindowListener(aSymWindow);
                SymAction lSymAction = new SymAction();
                //}}
        }

        public void setVisible(boolean b)
        {
                if (b)
                {
                        Rectangle bounds = (getParent()).getBounds();
                        Dimension size = getSize();
                        setLocation(bounds.x + (bounds.width - size.width)/2,
                                        bounds.y + (bounds.height - size.height)/2);
                }

                super.setVisible(b);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension d = getSize();

                super.addNotify();

                if (fComponentsAdjusted)
                        return;
                // Adjust components according to the insets
                Insets insets = getInsets();
                setSize(insets.left + insets.right + d.width, insets.top +
insets.bottom + d.height);
                Component components[] =
getContentPane().getComponents();
                for (int i = 0; i < components.length; i++)
                {
                        Point p = components[i].getLocation();
                        p.translate(insets.left, insets.top);
                        components[i].setLocation(p);
                }
                fComponentsAdjusted = true;
        }
```

```
        // Used for addNotify check.
        boolean fComponentsAdjusted = false;

        //{{DECLARE_CONTROLS
        //}}

        class SymWindow extends java.awt.event.WindowAdapter
        {
                public void windowClosing(java.awt.event.WindowEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JDialog_weight.this)
                                jAboutDialog_windowClosing(event);
                }
        }

        void jAboutDialog_windowClosing(java.awt.event.WindowEvent
event)
        {
                // to do: code goes here.

                jAboutDialog_windowClosing_Interaction1(event);
        }

        void
jAboutDialog_windowClosing_Interaction1(java.awt.event.WindowEvent
event) {
                try {
                        // JAboutDialog Hide the JAboutDialog
                        this.setVisible(false);
                } catch (Exception e) {System.out.println(e);
                }
        }

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                }
        }
}
```

```
package JobSchedule;

import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;

public class JDialog_weit extends com.sun.java.swing.JDialog
{

   public double w;
   public Weight weight;
   JFrame_manage p;
   int n;
        public JDialog_weit(Frame parent)
        {
                super(parent);
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Weight");
                getContentPane().setLayout(null);
                setSize(388,171);
                setVisible(false);
                JButton_delete_delcteperson.setText("Done");

        JButton_delete_deleteperson.setActionCommand("Delete");
                getContentPane().add(JButton_delete_deleteperson);
                JButton_delete_deleteperson.setBounds(156,132,76,24);
                JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel1.setText("Weight: ");
                getContentPane().add(JLabel1);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 16));
                JLabel1.setBounds(36,24,132,24);
                JLabel2.setText("(from 0.0 to 1.0)");
                getContentPane().add(JLabel2);
                JLabel2.setBounds(144,24,168,24);
                getContentPane().add(JTextField1);
                JTextField1.setBounds(96,84,192,24);
                getContentPane().add(JLabel3);
                JLabel3.setBounds(96,48,192,24);
                //}}

                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();

        JButton_delete_deleteperson.addActionListener(lSymAction);
                //}}
        }

        public JDialog_weit()
        {
                this((Frame)null);
        }

        public JDialog_weit(JFrame_manage p1, Weight weight1, int n1)
        {
                this((Frame)null);
                weight=weight1;
                p=p1;
                n=n1;
        }

        public JDialog_weit(String sTitle)
```

```
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new JDialog_weit()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets
                Insets insets = getInsets();
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JButton JButton_delete_deleteperson = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField JTextField1 = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JButton_delete_deleteperson)

        JButtonDeleteDeleteperson_actionPerformed(event);

                }
        }

        void
JButtonDeleteDeleteperson_actionPerformed(java.awt.event.ActionEvent
event)
        {
                // to do: code goes here.
                Double d=new
Double(((String)this.JTextField1.getText()).trim());
                w=d.doubleValue();
```

```
                    //int
i=Integer.parseInt(((String)this.JTextField1.getText()).trim());
                    if(0.0<=w&&w<=1.0){
                      weight.put_w(w);
                      if(n==1){
                        p.sortbyD_E();
                        p.datavalu();
                      }
                      else{
                         p.sortbyD_S();
                         p.datavalu();
                      }
                      this.setVisible(false);
        dispose();
        this.dispose();
                    }
                    else{
                      this.JLabel3.setText("Invalid Value");
                    }


          }
}
```



```
package JobSchedule;

import Cases.*;
import java.awt.*;
import com.sun.java.swing.*;
import java.util.*;

public class JFrame_assignjob extends com.sun.java.swing.JFrame
{
   Vector  person_queue,job_queue,work_queue;
          Vector   personskill_queue,jobskill_queue;
          Vector  result;
          int index, grade, step, n, year, month, date;
          String jobname;

          StepContent job;
          Vector p_q, pk_q, w_q, job_pool;
          CasesFrame parent;

          public JFrame_assignjob()
          {
                    result=new Vector();
                    step=1;
                    n=0;
                    // This code is automatically generated by Visual Cafe
when you add
                    // components to the visual environment. It instantiates
and initializes
                    // the components. To modify the code, only use code
syntax that matches
                    // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                    // parse your Java file into its visual environment.
                    //{{INIT_CONTROLS
                    setTitle("Job Assignment");
                    getContentPane().setLayout(null);
                    setSize(492,532);
```

```
            setVisible(false);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
            JLabel1.setText("Job Assignment");
            getContentPane().add(JLabel1);
            JLabel1.setFont(new Font("Dialog", Font.BOLD, 16));
            JLabel1.setBounds(150,12,192,24);

        JButton1.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.LEFT);
            JButton1.setText("       1. Filter by Security Level");
            JButton1.setActionCommand("by priority");
            getContentPane().add(JButton1);
            JButton1.setBounds(168,264,276,24);

        JButton2.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.LEFT);
            JButton2.setText("       2. Filter by Required Skills");
            JButton2.setActionCommand("by skills");
            getContentPane().add(JButton2);
            JButton2.setBounds(168,300,276,24);

        JButton3.setHorizontalAlignment(com.sun.java.swing.SwingConst
ants.LEFT);
            JButton3.setText("       3. Assign this Job");
            JButton3.setActionCommand("assign the job");
            getContentPane().add(JButton3);
            JButton3.setBounds(168,336,276,24);
            JButton4.setText("Exit");
            JButton4.setActionCommand("Save and Exit");
            getContentPane().add(JButton4);
            JButton4.setBounds(72,492,372,24);
            JScrollPane1.setOpaque(true);
            getContentPane().add(JScrollPane1);
            JScrollPane1.setBounds(72,372,372,108);
            JScrollPane1.getViewport().add(JTextArea2);
            JTextArea2.setBounds(0,0,369,105);
            JLabel2.setText("Security Level");
            getContentPane().add(JLabel2);
            JLabel2.setBounds(24,108,132,24);
            JLabel7.setText("Job ID");
            getContentPane().add(JLabel7);
            JLabel7.setBounds(24,72,84,28);
            JLabel8.setText("Deadline");
            getContentPane().add(JLabel8);
            JLabel8.setBounds(24,144,108,28);
            JLabel9.setText("Estimated Duration");
            getContentPane().add(JLabel9);
            JLabel9.setBounds(24,180,120,24);
            getContentPane().add(JTextField1);
            JTextField1.setBounds(168,180,276,28);
            getContentPane().add(JTextField2);
            JTextField2.setBounds(168,144,276,28);
            getContentPane().add(JTextField3);
            JTextField3.setBounds(168,108,276,28);
            getContentPane().add(JTextField4);
            JTextField4.setBounds(168,72,276,28);
            JLabel3.setText("Required Skills");
            getContentPane().add(JLabel3);
            JLabel3.setBounds(24,216,120,24);
            JButton5.setText("Required Skills");
            JButton5.setActionCommand("Job skill");
            getContentPane().add(JButton5);
            JButton5.setBounds(168,216,276,24);
            //}}

            //{{INIT_MENUS
            //}}

            //{{REGISTER_LISTENERS
            SymAction lSymAction = new SymAction();
            JButton4.addActionListener(lSymAction);
            JButton1.addActionListener(lSymAction);
```

```
                JButton2.addActionListener(lSymAction);
                JButton3.addActionListener(lSymAction);
                JButton5.addActionListener(lSymAction);
                //}}

                //{{REGISTER_LISTENERS
                //SymAction lSymAction = new SymAction();
                //JButton3.addActionListener(lSymAction);
                //}}


        }

        public JFrame_assignjob(CasesFrame parent1,Vector
person_queue1,Vector job_queue1, Vector job_pool1)
        {
                this();

                parent=parent1;
                person_queue=person_queue1; job_queue=job_queue1;
                job_pool=job_pool1;
                this.job=(StepContent) job_queue.firstElement();
                String name=job.getStepName();
                this.jobname=name;


                String sk1=new String();
                this.JTextField4.setText(job.getStepName());

        this.JTextField3.setText(sk1.valueOf(job.getSecurityLevel()));
                this.JTextField2.setText(job.getDeadline());
                this.JTextField1.setText(sk1.valueOf(job.getDuration()));
                //System.out.println("in assignment constructor person
size:"+person_queue.size());
      //cleanperson_job();
        }

        public JFrame_assignjob(String sTitle)
        {
                this();
                setTitle(sTitle);

        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new JFrame_assignjob()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
```

```
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton JButton1 = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton JButton2 = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton JButton3 = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JButton JButton4 = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JScrollPane JScrollPane1 = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JTextArea JTextArea2 = new
com.sun.java.swing.JTextArea();
        com.sun.java.swing.JLabel JLabel2 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel7 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel8 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JLabel JLabel9 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JTextField JTextField1 = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField JTextField2 = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField JTextField3 = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JTextField JTextField4 = new
com.sun.java.swing.JTextField();
        com.sun.java.swing.JLabel JLabel3 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton JButton5 = new
com.sun.java.swing.JButton();
        //}}

        //{{DECLARE_MENUS
        //}}


        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JButton4)
                                JButton4_actionPerformed(event);
                        else if (object == JButton1)
                                JButton1_actionPerformed(event);
                        else if (object == JButton2)
                                JButton2_actionPerformed(event);
                        else if (object == JButton3)
                                JButton3_actionPerformed(event);
                        else if (object == JButton5)
                                JButton5_actionPerformed(event);
                }
        }

        void JButton4_actionPerformed(java.awt.event.ActionEvent event)
        {
                // to do: code goes here.
                this.setVisible(false);
```

```
        }

        void Filterbysecurity(){

          // p_q=(Vector) person_queue.clone();
           p_q=new Vector();
           int security=job.getSecurityLevel();
           System.out.println("job security is "+security);
           for(int i=0; i<person_queue.size(); i++){
              Personnel person=(Personnel) person_queue.elementAt(i);
              System.out.println("the person security is
"+person.getSecurityLevel());
              if(security<=person.getSecurityLevel()){
                 System.out.println("the security is
"+person.getSecurityLevel()
                 +"persion id is "+person.getID());

                 p_q.addElement(person);

              }
           }//end for
           System.out.println("in filterbysecurity: p_q size is "+p_q.size());
        }//end Filter


        int getmonthbyint(String month){
           if(month.equals("January")){
              return 1;
           }
           else{
              if(month.equals("February")){
                 return 2;
              }
              else{
                 if(month.equals("March")){
                    return 3;
```

```
}
else{
   if(month.equals("April")){
      return 4;
   }
   else{
      if(month.equals("May")){
         return 5;
      }
      else{
         if(month.equals("June")){
            return 6;
         }
         else{
            if(month.equals("July")){
               return 7;
            }
            else{
               if(month.equals("August")){
                  return 8;
               }
               else{
                  if(month.equals("September")){
                     return 9;
                  }
                  else{
                     if(month.equals("October")){
                        return 10;
                     }
                     else{
                        if(month.equals("November")){
                           return 11;
                        }
                        else{
                           if(month.equals("December")){
                              return 12;
                           }
```

```
                    }
                }
            }
        }
    }
}
}//end if-else
return 0;
}

String setmonth(int month){
if(month==1){
String s="January";
return s;
}
else{
if(month==2){
String s="February";
return s;
}
else{
if(month==3){
String s="March";
return s;
}
else{
if(month==4){
String s="April";
return s;
}
else{
if(month==5){
String s="May";
return s;
}
else{
if(month==6){
String s="June";
return s;
}
else{
if(month==7){
String s="July";
return s;
}
else{
if(month==8){
String s="August";
return s;
}
else{
if(month==9){
String s="September";
return s;
}
else{
if(month==10){
String s="October";
return s;
}
else{
if(month==11){
String s="November";
return s;
}
else{
if(month==12){
String s="December";
return s;
}
```

```
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return null;

}//end

void getdate(String date){

    this.year=0; this.month=0; this.date=0;
    System.out.println("in getdate");
    StringTokenizer st = new StringTokenizer(date,",");
    st.hasMoreTokens();
            String s=new String(st.nextToken());
            String s2=new String(st.nextToken());

            StringTokenizer st1 = new StringTokenizer(s," ");
            st1.hasMoreTokens();
            String s1=st1.nextToken();
            this.month=getmonthbyint(s1);
            try{
               this.date=Integer.parseInt(st1.nextToken());
               //this.year=Integer.parseInt(s);
}
catch( Exception e){}

}

void Filterbyskills(){

            try{
               for(int i=0; i<this.p_q.size(); i++){
                  this.grade=0;
                  Personnel p=(Personnel) p_q.elementAt(i);
                  Vector v=(Vector) p.getSkill();

                  for(int j=0; j<v.size(); j++){
                     String s1=(String)v.elementAt(j);

                     StringTokenizer st = new StringTokenizer(s1,":");
                     String s2=new String(st.nextToken());
                           int number1=Integer.parseInt(s2.trim());
                           String name1=((String)st.nextToken()).trim();
                           int
level1=Integer.parseInt(((String)st.nextToken()).trim());
                           System.out.println("take each the person's skill");
                           System.out.println("the skill number, name,
level:");
                           System.out.println(number1+name1+level1);
                           for(int k=0; k<this.job.getSkill().size(); k++){
                              String
s3=(String)this.job.getSkill().elementAt(k);
                              System.out.println("the job skill is "+s3);
                              StringTokenizer st2 = new
StringTokenizer(s3,":");
                              String s4=((String)(st2.nextToken())).trim();
                              int number2=Integer.parseInt(s4.trim());
                              String name2=((String)st2.nextToken()).trim();
                              int
level2=Integer.parseInt(((String)st2.nextToken()).trim());
                              System.out.println("the job's number, name,
level:");
                              System.out.println(number2+" "+name2+"
"+level2);
                              if(number1==number2&&level1>=level2){
                                 grade++;
```

```
                        System.out.println("number1,
level1"+number1+level1);
                        System.out.println("number2,
level2"+number2+level2);
                    }
                }//end for(k)

            }//end for(j)
            int id=Integer.parseInt(p.getID());
            Result r=new Result(id,grade);
            result.addElement(r);

        }//end for(i)
        sortresult();
    }
    catch( Exception e){}
    }

    void sortresult(){

        Result min, tmp;
        for(int i=0; i<result.size(); i++){
            min=(Result) result.elementAt(i);
            for(int j=i+1; j<result.size(); j++){
                Result r1=(Result) result.elementAt(j);

                if(min.get_grade()<r1.get_grade()){
                    tmp=min;
                    min=r1;
                    result.setElementAt(tmp, j);
                }//end if()
            }//end for(j)

            result.setElementAt(min, i);
        }//end for(//i)
    }//end sort

    Personnel seachbyid(int id){
        try{
            for(int j=0; j<p_q.size(); j++){
                Personnel p=(Personnel) p_q.elementAt(j);
                int id1=Integer.parseInt(p.getID());
                if(id==id1){
                    return p;
                }
            }
    }
    catch( Exception e){}
        return null;
    }//end seach

    void display1(){

        this.JTextArea2.setText("");
        System.out.println("p_q siz is "+p_q.size());
        this.JTextArea2.append("Stakeholder ID"+"\t"+"Security
Level"+"\n");
                String sk=new String();
                for(int i=0; i<p_q.size(); i++){
                    Personnel p=(Personnel) p_q.elementAt(i);
            this.JTextArea2.append(sk.valueOf(p.getID()+"\t"));

this.JTextArea2.append(sk.valueOf(p.getSecurityLevel()+"\n"));

        }

    }

    void display2(){

        this.JTextArea2.setText("");
```

```
            this.JTextArea2.append("Stakeholder ID"+"\t"+"Match Number
of Required Skills"+"\n");
            String s=new String();

           for(int i=0; i<result.size(); i++){
                    Result r=(Result) result.elementAt(i);
              this.JTextArea2.append(s.valueOf(r.get_id()+"\t"));
              this.JTextArea2.append(s.valueOf(r.get_grade()+"\n"));

            }

          }

          void display3(int pid){
            Personnel p=seachbyid(pid);
            Vector v=p.getMajorJobs();

            StepContent step1=(StepContent) v.elementAt(0);
            StepContent step2=null;
            if(v.size()==2){
              step2=(StepContent) v.elementAt(1);
            }

            this.JTextArea2.setText("");
            this.JTextArea2.append("Stakeholder ID"+"\t"+"Job
ID"+"\t"+"Earliest Start Time"
                              +"\t"+"Estimated Duration"+"\n");
            this.JTextArea2.append("");
            this.JTextArea2.append(p.getID()+"\t"+
step1.getStepName()+"\t"+step1.getStartTime()+"\t"+step1.getDuration()
+"\n");
            if(step2!=null){

this.JTextArea2.append(p.getID()+"\t"+step2.getStepName()+"\t"+step2.get
StartTime()+"\t"+step2.getDuration()+"\n");
            }
          }


          void JButton1_actionPerformed(java.awt.event.ActionEvent event)
          {
                  // to do: code goes here.
                  if(step==1){
                    step++;
                    Filterbysecurity();
                    display1();
                  }


          }

          void JButton2_actionPerformed(java.awt.event.ActionEvent event)
          {
                  // to do: code goes here.
                  if(step==2){
                    step++;
                    Filterbyskills();
                    display2();
                  }

            symantec.itools.awt.util.Calendar c=new
symantec.itools.awt.util.Calendar();
            String s=c.getDate();
            System.out.println("today is "+s);
            getdate(s);
            System.out.println("year, month, date are:");
            System.out.println(year+" "+ month+" "+ date);
          }

          void JButton3_actionPerformed(java.awt.event.ActionEvent event)
          {
                  // to do: code goes here.
                  if(step==3||step==4){
```

```
                    step++;
                    int  num=0;
                    if(n==0){

                       num=assing_Performed();
                       if(num==1){
                          System.out.println("job jobname is "+jobname);
                          System.out.println("befor save job_pool size=
"+job_pool.size());
                          for(int i=0; i<job_pool.size(); i++){
                             StepContent
step=(StepContent)job_pool.elementAt(i);
                         System.out.println("before save the step status and
name  "+step.getStatus()+"  "+step.getStepName());
                    }
                          parent.savePersonnelVector(person_queue);
                          parent.saveStepContentVector(job_pool);
                       }
                       n++;
                     }
                   }
          }//end

          int assing_Performed(){

             symantec.itools.awt.util.Calendar c=new
symantec.itools.awt.util.Calendar();
             int duration, personid;
             String jobname;
             Result r;

             duration=this.job.getDuration();
             jobname=this.job.getStepName();

             for(int i=0; i<result.size(); i++){

               personid=((Result) result.elementAt(i)).get_id();

Personnel p=seachbyid(personid);
Vector v1=p.getMajorJobs();
Vector v2=p.getMinorJobs();

String s=c.getDate();
System.out.println("clear person");

if(v1.size()<2){
 //set the job's status
 job.setStatus("Assigned");
 //set the job's starttime
 job.setStartTime(s);
 //save job
 setjob_pool(jobname, this.job);
 //set the person's morja job
 ((StepContent)this.job).setRealStartTime(c.getDate());
 v1.addElement(this.job);
 p.setMajorJobs(v1);
 //save the person
 setperson_queue(personid, p);

 //remove the job
 job_queue.removeElement(this.job);

 display3(personid);

 return 1;
}//end if
else{

 //set the person's minor job
 ((StepContent)this.job).setRealStartTime(c.getDate());
 v2.addElement(this.job);
 //set Mina job
 p.setMinorJobs(v2);
 //save the person
 setperson_queue(personid, p);
```

```
            return 1;
            }

          }//end for

          return 0;

        }//end assing

        void setperson_queue(int personid, Personnel p1){
          try{
            for(int i=0; i<person_queue.size(); i++){
              Personnel p=(Personnel) person_queue.elementAt(i);
              if(personid==Integer.parseInt(p.getID())){
                person_queue.setElementAt(p1, i);
              }
            }//end for_loop
          }
      catch(Exception e){}
        }
        void setjob_pool(String jobname, StepContent step1){
          for(int i=0; i<job_pool.size(); i++ ){
            StepContent step=(StepContent) job_pool.elementAt(i);
            if(jobname.equals(step.getStepName())){
              job_pool.setElementAt(step1, i);
            }
          }
        }
        void cleanperson_job(){
          try{
            for(int i=0; i<person_queue.size(); i++){
              Personnel p=(Personnel) person_queue.elementAt(i);
              Vector v1=p.getMajorJobs();
              Vector v2=p.getMinorJobs();


              v1.removeAllElements();
                p.setMajorJobs(v1);
                v2.removeAllElements();
                p.setMinorJobs(v2);
                int personid=Integer.parseInt(p.getID());
                setperson_queue(personid, p);
              }
              parent.savePersonnelVector(person_queue);
        }
        catch( Exception e){}
          }
          void JButton5_actionPerformed(java.awt.event.ActionEvent event)
          {
                // to do: code goes here.
                Vector v=(Vector) job.getSkill();
                (new JDialog_jobskill(v)).setVisible(true);


          }
}
```

```
package JobSchedule;

/*
        A basic implementation of the JFrame class.
*/
import Cases.*;
import java.util.*;
import java.util.Date;
import java.awt.*;
import com.sun.java.swing.*;
import com.sun.java.swing.table.DefaultTableModel;
import java.lang.Double;
import com.symantec.itools.swing.JButtonGroupPanel;

public class JFrame_manage extends com.sun.java.swing.JFrame
{
   Vector job_pool, job_queue1,job_queue2, job_queue;
   double w;
   Weight weight;
   //TableModel mytable;
   int year=0, month=0, date=0;

        public JFrame_manage()
        {
           weight=new Weight();
           job_queue1=new Vector();
           job_queue2=new Vector();

           //initai weight value
           w=0.5;
           System.out.println("in manage2");
                // This code is automatically generated by Visual Cafe
when you add
                // components to the visual environment. It instantiates
and initializes
                // the components. To modify the code, only use code
syntax that matches
                // what Visual Cafe can generate, or Visual Cafe may be
unable to back
                // parse your Java file into its visual environment.
                //{{INIT_CONTROLS
                setTitle("Job Scheduling");
                getContentPane().setLayout(null);
                setSize(596,415);
                setVisible(false);

        JLabel1.setHorizontalAlignment(com.sun.java.swing.SwingConsta
nts.CENTER);
                JLabel1.setText("Job Scheduling Policy");
                getContentPane().add(JLabel1);
                JLabel1.setFont(new Font("Dialog", Font.BOLD, 16));
                JLabel1.setBounds(190,24,216,36);
                JButton5.setText("Exit");
                JButton5.setActionCommand("Save and Exit");
                getContentPane().add(JButton5);
                JButton5.setBounds(256,372,92,24);
                getContentPane().add(JLabel6);
                JLabel6.setBounds(24,72,108,24);
                JScrollPane2.setOpaque(true);
                getContentPane().add(JScrollPane2);
                JScrollPane2.setBounds(60,132,480,216);
                JScrollPane2.getViewport().add(JTable1);
                JTable1.setBounds(0,0,477,213);
                getContentPane().add(JComboBox1);
                JComboBox1.setBounds(144,72,396,36);
                //}}

                //{{INIT_MENUS
                //}}



                //{{REGISTER_LISTENERS
                SymAction lSymAction = new SymAction();
```

```
                JButton5.addActionListener(lSymAction);
                SymFocus aSymFocus = new SymFocus();
                SymItem lSymItem = new SymItem();
                JComboBox1.addItemListener(lSymItem);
                JScrollPane2.addFocusListener(aSymFocus);
                JTable1.addFocusListener(aSymFocus);
                //}}
                JComboBox1.addItem("Select a Policy");
                JComboBox1.addItem("High Priority First");
                JComboBox1.addItem("Minimum Deadline First
(Min_D)");
                JComboBox1.addItem("Minimum Estimated Duration
First (Min_E)");
                JComboBox1.addItem("Minimum Earliest Start Time
First (Min_S)");
                JComboBox1.addItem("Minimum Laxity First
(Min_L)");
                JComboBox1.addItem("Min_D*w + Min_E*(1-w)");
                JComboBox1.addItem("Min_D*w + Min_S*(1-w)");

        }

        public JFrame_manage(Vector job_queue1)
        {
                this();
                System.out.println("this is in manage");
                job_queue=job_queue1;
                System.out.println("job_queue="+job_queue.size());
                datavalu();
                //display();
        }

        public JFrame_manage(String sTitle)
        {
                this();
                setTitle(sTitle);
        }

        public void setVisible(boolean b)
        {
                if (b)
                        setLocation(50, 50);
                super.setVisible(b);
        }

        static public void main(String args[])
        {
                (new JFrame_manage()).setVisible(true);
        }

        public void addNotify()
        {
                // Record the size of the window prior to calling parents
addNotify.
                Dimension size = getSize();

                super.addNotify();

                if (frameSizeAdjusted)
                        return;
                frameSizeAdjusted = true;

                // Adjust size of frame according to the insets and menu
bar
                Insets insets = getInsets();
                com.sun.java.swing.JMenuBar menuBar =
getRootPane().getJMenuBar();
                int menuBarHeight = 0;
                if (menuBar != null)
                        menuBarHeight =
menuBar.getPreferredSize().height;
                setSize(insets.left + insets.right + size.width, insets.top +
insets.bottom + size.height + menuBarHeight);
```

```
        }

        // Used by addNotify
        boolean frameSizeAdjusted = false;

        //{{DECLARE_CONTROLS
        com.sun.java.swing.JLabel JLabel1 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JButton JButton5 = new
com.sun.java.swing.JButton();
        com.sun.java.swing.JLabel JLabel6 = new
com.sun.java.swing.JLabel();
        com.sun.java.swing.JScrollPane JScrollPane2 = new
com.sun.java.swing.JScrollPane();
        com.sun.java.swing.JTable JTable1 = new
com.sun.java.swing.JTable();
        com.sun.java.swing.JComboBox JComboBox1 = new
com.sun.java.swing.JComboBox();
        //}}

        //{{DECLARE_MENUS
        //}}

        //creat Jtable
        void datavalu(){

          DefaultTableModel mymodel;
          int size=job_queue.size();
          String datavalue[][]=new String[size][6];
    String header[]=new String[6];

          header[0]="Job ID";
          header[1]="Priority";
          header[2]="Deadline";
          header[3]="Estimated Duration";
          header[4]="Earliest Starttime";
          header[5]="Laxity";

                for(int i=0; i<size; i++){
                  String s=new String();
                    StepContent step=(StepContent)job_queue.elementAt(i);
                    datavalue[i][0]=step.getStepName();
                    datavalue[i][1]=s.valueOf(step.getPriority());
                    datavalue[i][2]=step.getDeadline();
                    datavalue[i][3]=s.valueOf(step.getDuration());
                    datavalue[i][4]=step.getStartTime();

                    //caculation laxity
                    int laxity, y1,y2,m1,m2,d1,d2;

                    String deadline=step.getDeadline();
                    getdate(deadline);
                    y1=0;  m1=0;  d1=0;
                    y1=this.year;  m1=this.month;  d1=this.date;

                    String starttime=step.getStartTime();
                    getdate(starttime);
                    y2=0;  m2=0;  d2=0;
                    y2=this.year;  m2=this.month;  d2=this.date;
                    laxity=((y1-y2)*360+(m1-m2)*30+(d1-d2))-
step.getDuration();

                    datavalue[i][5]=s.valueOf(laxity);


                }
                mymodel=new DefaultTableModel(datavalue, header);
                JTable1.setModel(mymodel);
                System.out.println("after creat JTable");

        }


        int getmonthbystring(String month){
```

```
if(month.equals("January")){
  return 1;
}
else{
  if(month.equals("February")){
    return 2;
  }
  else{
    if(month.equals("March")){
      return 3;
    }
    else{
      if(month.equals("April")){
        return 4;
      }
      else{
        if(month.equals("May")){
          return 5;
        }
        else{
          if(month.equals("June")){
            return 6;
          }
          else{
            if(month.equals("July")){
              return 7;
            }
            else{
              if(month.equals("August")){
                return 8;
              }
              else{
                if(month.equals("September")){
                  return 9;
                }
                else{
                  if(month.equals("October")){
                    return 10;
                  }
                  else{
                    if(month.equals("November")){
                      return 11;
                    }
                    else{
                      if(month.equals("December")){
                        return 12;
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }//end if-else
  return 0;
}

String getmonthbyint(int month){
  if(month==1){
    String s="January";
    return s;
  }
  else{
    if(month==2){
      String s="February";
      return s;
    }
    else{
      if(month==3){
        String s="March";
```

```
        return s;
    }
    else{
        if(month==4){
            String s="April";
            return s;
        }
        else{
            if(month==5){
                String s="May";
                return s;
            }
            else{
                if(month==6){
                    String s="June";
                    return s;
                }
                else{
                    if(month==7){
                        String s="July";
                        return s;
                    }
                    else{
                        if(month==8){
                            String s="August";
                            return s;
                        }
                        else{
                            if(month==9){
                                String s="September";
                                return s;
                            }
                            else{
                                if(month==10){
                                    String s="October";
                                    return s;
                                }
                                else{
                                    if(month==11){
                                        String s="November";
                                        return s;
                                    }
                                    else{
                                        if(month==12){
                                            String s="December";
                                            return s;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return null;

}//end

void sortbypriority(){

    StepContent max, tmp;
    for(int i=0; i<job_queue.size(); i++){
        max=(StepContent) job_queue.elementAt(i);
        for(int j=i+1; j<job_queue.size(); j++){
            StepContent step1=(StepContent) job_queue.elementAt(j);
            if(max.getPriority()<step1.getPriority()){
                tmp=max;
                max=step1;
                job_queue.setElementAt(tmp, j);
            }//end if()
```

```
      }//end for(j)

      job_queue.setElementAt(max, i);
    }//end for(//i)

}//end sort

void sortbydeadline(){

System.out.println("in sorbydeadline ");

    StepContent earlist, tmp;
    int y1=0, y2=0, m1=0, m2=0, d1=0, d2=0;
    for(int i=0; i<job_queue.size(); i++){
      earlist=(StepContent) job_queue.elementAt(i);

      String dead1=earlist.getDeadline();

      getdate(dead1);
      y1=0; m1=0; d1=0;
      y1=this.year;  m1=this.month;  d1=this.date;

      for(int j=i+1; j<job_queue.size(); j++){
        StepContent step1=(StepContent) job_queue.elementAt(j);
        String dead2=step1.getDeadline();
        getdate(dead2);
        y2=0; m2=0; d2=0;
        y2=this.year;  m2=this.month;  d2=this.date;

        int n=after(y1,m1,d1,y2,m2,d2);

        if(n==1){
          dead1=step1.getDeadline();
          getdate(dead1);
          y1=0; m1=0; d1=0;
          y1=this.year;  m1=this.month;  d1=this.date;
          tmp=earlist;
          earlist=step1;
          job_queue.setElementAt(tmp, j);
        }//end if()
      }//end for(j)

      job_queue.setElementAt(earlist, i);
    }//end for(//i)

}

int after(int y1, int m1, int d1, int y2, int m2, int d2){
    if(y1>y2){
      return 1;
    }
    else{
      if(y1==y2){
        if(m1>m2){
          return 1;
        }
        else{
          if(m1==m2){
            if(d1>d2){
              return 1;
            }
          }
        }
      }
    }
    return 0;
}//end after

void getdate(String date){
    try{
      this.year=0; this.month=0; this.date=0;
      StringTokenizer st = new StringTokenizer(date," ");
      st.hasMoreTokens();
          String s=new String(st.nextToken());
```

```
            this.month=getmonthbystring(s);
            String d=st.nextToken();
            this.year=Integer.parseInt(st.nextToken());

            StringTokenizer st1 = new StringTokenizer(d,",");
        st1.hasMoreTokens();
            String d2=new String(st1.nextToken());
            this.date=Integer.parseInt(d2);
}
catch(Exception e){}
    }


    void sortbyduration(){

      StepContent min, tmp;
      int y1, y2, m1, m2, d1, d2;
      for(int i=0; i<job_queue.size(); i++){
        min=(StepContent) job_queue.elementAt(i);
        for(int j=i+1; j<job_queue.size(); j++){
          StepContent step1=(StepContent) job_queue.elementAt(j);
          if(min.getDuration()>step1.getDuration()){
            tmp=min;
            min=step1;
            job_queue.setElementAt(tmp, j);
          }//end if()
        }//end for(j)

        job_queue.setElementAt(min, i);
      }//end for(//i)

    }

    void sortbystarttime(){

      int y1, y2, m1, m2, d1, d2;

      StepContent ealist, tmp;
      for(int i=0; i<job_queue.size(); i++){
        ealist=(StepContent) job_queue.elementAt(i);
        String dead1=ealist.getStartTime();
        getdate(dead1);
        y1=this.year;  m1=this.month;  d1=this.date;

        for(int j=i+1; j<job_queue.size(); j++){
          StepContent step1=(StepContent) job_queue.elementAt(j);
          String dead2=step1.getStartTime();
          getdate(dead2);
           y2=0; m2=0; d2=0;
          y2=this.year;  m2=this.month;  d2=this.date;

          if(after(y1,m1,d1,y2,m2,d2)==1){
            dead1=step1.getStartTime();
            getdate(dead1);
            y1=this.year;  m1=this.month;  d1=this.date;
            tmp=ealist;
            ealist=step1;
            job_queue.setElementAt(tmp, j);
          }//end if()
        }//end for(j)

        job_queue.setElementAt(ealist, i);
      }//end for(//i)

}//end sortbystarttime

void sortbylaxity(){

      StepContent min, tmp;
      int laxity1,laxity2;
      int y1, y2, m1, m2, d1, d2;
      for(int i=0; i<job_queue.size(); i++){
        min=(StepContent) job_queue.elementAt(i);
        String deadline1=min.getDeadline();
```

```
            getdate(deadline1);
            y1=0; m1=0; d1=0;
            y1=this.year; m1=this.month; d1=this.date;

            String starttime1=min.getStartTime();
            getdate(starttime1);
            y2=0; m2=0; d2=0;
            y2=this.year; m2=this.month; d2=this.date;
            laxity1=((y1-y2)*360+(m1-m2)*30+(d1-d2))-
min.getDuration();

            for(int j=i+1; j<job_queue.size(); j++){
              StepContent step=(StepContent) job_queue.elementAt(j);
              String deadline2=step.getDeadline();
              getdate(deadline2);
              y1=0; m1=0; d1=0;
              y1=this.year; m1=this.month; d1=this.date;
              String starttime2=step.getStartTime();
              getdate(starttime2);
              y2=0; m2=0; d2=0;
              y2=this.year; m2=this.month; d2=this.date;
              laxity2=((y1-y2)*360+(m1-m2)*30+(d1-d2))-
step.getDuration();

              if(laxity1>laxity2){

                tmp=min;
                min=step;
                job_queue.setElementAt(tmp, j);
              }//end if()
            }//end for(j)
            job_queue.setElementAt(min, i);
          }//end for(//i)

        }//end sortbylaxity()

        //sortbyMinD_MinE
```

```
void sortbyD_E(){

  sortbydeadline();
  System.out.println("in D_E e="+w);
  w=weight.get_w();
  StepContent step=(StepContent) job_queue.lastElement();
  String s=step.getDeadline();
  getdate(s);
  int y=year;
  int m=month;
  int d=date;
  System.out.println("month= "+m+"  date= "+d);

  StepContent max, tmp;

  for(int i=0; i<job_queue.size(); i++){
    max=(StepContent) job_queue.elementAt(i);
    String deadline1=max.getDeadline();
    getdate(deadline1);
    int y_dead1=this.year;
    int m_dead1=this.month;
    int d_dead1=this.date;
    for(int j=i+1; j<job_queue.size(); j++){
      StepContent step1=(StepContent) job_queue.elementAt(j);
      String deadline2=step1.getDeadline();
      getdate(deadline2);
      int y_dead2=year;
      int m_dead2=month;
      int d_dead2=date;
      if((((y-y_dead1)*360+(m-m_dead1)*30+(d-d_dead1))*w
          -max.getDuration()*(1-w))<
        (((y-y_dead2)*360+(m-m_dead2)*30+(d-d_dead2))*w
        -step1.getDuration()*(1-w)))
      {
        tmp=max;
        max=step1;
        job_queue.setElementAt(tmp, j);
```

```
        }//end if()
      }//end for(j)


      job_queue.setElementAt(max, i);
    }//end for(//i)



}

void sortbyD_S(){

   sortbydeadline();
   System.out.println("in D_S e="+w);

   w=weight.get_w();
   StepContent step=(StepContent) job_queue.lastElement();
   //int y=job.get_deadline().getYear();
   String s1=step.getDeadline();
   getdate(s1);
   int y_dead=year;
   int m_dead=month;
   int d_dead=date;

   String s2=step.getStartTime();
   getdate(s2);
   int y_start=year;
   int m_start=month;
   int d_start=date;

   StepContent max, tmp;
   for(int i=0; i<job_queue.size(); i++){
      max=(StepContent) job_queue.elementAt(i);

      String s3=max.getDeadline();
                  getdate(s3);
                  int y_dead_max=year;
                  int m_dead_max=month;
                  int d_dead_max=date;

                  String s4=max.getStartTime();
                  getdate(s4);
                  int y_start_max=year;
                  int m_start_max=month;
                  int d_start_max=date;

                  for(int j=i+1; j<job_queue.size(); j++){
                     StepContent step1=(StepContent) job_queue.elementAt(j);

                     String s5=step1.getDeadline();
                     getdate(s5);
                     int y_dead_step=year;
                     int m_dead_step=month;
                     int d_dead_step=date;

                     String s6=step1.getStartTime();
                     getdate(s6);
                     int y_start_step=year;
                     int m_start_step=month;
                     int d_start_step=date;

                     if((((y_dead-y_dead_max)*365+(m_dead-
m_dead_max)*30+(d_dead-d_dead_max))*w
                           +((y_start-y_start_max)*365+(m_start-
m_start_max)*30+(d_start-d_start_max))*(1-w))<
                           (((y_dead-y_dead_step)*365+(m_dead-
m_dead_step)*30+(d_dead-d_dead_step))*w
                           +((y_dead-y_start_step)*365+(m_dead-
m_start_step)*30+(d_dead-d_start_step))*(1-w)))
                        {
                           tmp=max;
                           max=step1;
```

```
                job_queue.setElementAt(tmp, j);

              }//end if()
            }//end for(j)

            job_queue.setElementAt(max, i);
          }//end for(//i)

        }

        class SymAction implements java.awt.event.ActionListener
        {
                public void actionPerformed(java.awt.event.ActionEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JButton5)
                                JButton5_actionPerformed(event);




                }
        }

        void JButton5_actionPerformed(java.awt.event.ActionEvent event)
        {
                // to do: code goes here.
                this.setVisible(false);

        }

        class SymFocus extends java.awt.event.FocusAdapter
        {
                public void focusGained(java.awt.event.FocusEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JScrollPane2)
                                JScrollPane2_focusGained(event);
                        else if (object == JTable1)
                                JTable1_focusGained(event);
                }
        }

        class SymItem implements java.awt.event.ItemListener
        {
                public void itemStateChanged(java.awt.event.ItemEvent
event)
                {
                        Object object = event.getSource();
                        if (object == JComboBox1)

        JComboBox1_itemStateChanged(event);

                }
        }

        void JComboBox1_itemStateChanged(java.awt.event.ItemEvent
event)
        {
          System.out.println("in jombobox performence");

    if( event.getStateChange() == event.SELECTED ){
                        // to do: code goes here.
                        String str=(String) event.getItem();

                        if(str.equals("High Priority First")){
                          System.out.println("high priority first");
                          //new Getdat();
                          sortbypriority();
                          datavalu();
                        }
                        else{
```

```
                if(str.equals("Minimum Deadline First (Min_D)")){
                System.out.println("deadline");
                sortbydeadline();
                datavalu();
                }
                else
                  if(str.equals("Minimum Estimated Duration First
(Min_E)")){
                  System.out.println("duration");
                  sortbyduration();
                  datavalu();
                  }
                  else{
                    if(str.equals("Minimum Earliest Start Time First
(Min_S)" )){
                    System.out.println("startime");
                    sortbystarttime();
                    datavalu();
                    }
                    else{
                      if(str.equals("Minimum Laxity First
(Min_L)")){
                      System.out.println("min_l");
                          sortbylaxity();
                          datavalu();
                          }
                          else{
                            if(str.equals("Min_D*w + Min_E*(1-
w)")){
                        System.out.println("D_E");
                        (new JDialog_weit(this, weight,
1)).setVisible(true);
                          //sortbyD_E();
                        }
                        else{
                          if(str.equals("Min_D*w + Min_S*(1-
w)")){
                                System.out.println("D_S");
                                (new JDialog_weit(this, weight,
2)).setVisible(true);
                                //sortbyD_S();
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
            }
            void JScrollPane2_focusGained(java.awt.event.FocusEvent event)
            {
                // to do: code goes here.

            }
            void JTable1_focusGained(java.awt.event.FocusEvent event)
            {
                // to do: code goes here.

            }
        }
```

```
package JobSchedule;

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import java.io.Serializable;


public class Predecessor implements Serializable{
   private Vector predec;
   public Predecessor(){predec=new Vector();}
   public Vector get_predec(){ return predec;}
}
```

```
package JobSchedule;

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import java.io.Serializable;
import com.sun.java.swing.*;

public class Result implements Serializable{
   private int personid;
   private int grade;
   public Result(int id, int grade1){
      personid=id;
      grade=grade1;
   }

   public Result() {}
   public int get_id(){ return personid; }
   public int get_grade(){ return grade; }
}
```

```
package JobSchedule;

import java.awt.*;
import java.io.Serializable;

public class Weight implements Serializable{
   private double w;
   private int i;
   public Weight(){ w=0.5; i=5;}
   public Weight(double w1){    w=w1;
   }
   public void put_w(double w1){w=w1;}
   public void put_i(int i1){i=i1;}
   public double get_w(){return w;}
   public int get_i(){return i;}
}
```

```
package JobSchedule;

import java.util.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import java.io.Serializable;


public class Work_schedul implements Serializable{
   private int personid;
   private Vector schedul;
   public Work_schedul(int id){
      personid=id;
      schedul=new Vector();
   }

   public Work_schedul() {}
   public int get_id(){ return personid;}
   public Vector get_schedul(){ return schedul;}
}
```

LIST OF REFERENCES

[BADR94] S. Badr and Luqi, "Automation Support for Concurrent Software Engineering," *Proceeding of the 6th International Conference on Software Engineering and Knowledge Engineering*, Jurmala, Latvia, June 20-23, 1994, pp. 46-53.

[BERG89] C. Berge, *Hypergraphs*, North-Holland, 1989.

[BERZ91] V. Berzins and Luqi, *Software Engineering with Abstractions*, Addison-Wesley, 1991.

[BERZ97] V. Berzins, O. Ibrahim, and Luqi (1997), "A Requirements Evolution Model for Computer Aided Prototyping", *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering*, Madrid, Spain, June 17-20, 1997, pp. 38-47.

[BERZ98] V. Berzins, "Lightweight Inference for Automation Efficiency", *Proceedings of the 1998 ARO/ONR/NSF/DARPA Monterey Workshop on Engineering Automation for computer Based Systems*, Carmel, California, October 23-26, 1998, pp. 19-29.

[CONK88] J. Conklin and M. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information System*, Vol. 6, October 1988, pp. 303-331.

[DAMP94] D. A. Dampier, Luqi, and V. Berzins, "Automated Merging of Software Prototypes," *Journal of Systems Integration*, Vol. 4, No. 1, February 1994, pp. 33-49.

[DATT99] S. Datta, C. Weaver, D. Parfitt, and M. Rothstein, "Systems Engineering Life Cycle Management with Traceability," *Proceedings of the Thirteenth International Conference on Systems Engineering*, Las Vegas, Nevada, August 9-12, 1999, pp. SE: 61-66.

[EVAN97] J. Evans, "Project Scheduling Tool," *Master Thesis*, Computer Science Department, Naval Postgraduate School, Monterey, CA, 1997.

[FLAN97] D. Flanagan, "Java in a Nutshell", Second Edition, O'Reilly & Associates, Inc., 1997.

[GALL93] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed Hypergraphs and Applications," *Discrete Applied Mathematics* 42, 1993, pp. 177-201.

[GOGU96] J. Goguen, D. Nguyen, J. Meseguer, Luqi, D. Zhang, and V. Berzins, "Software Component Search," *Journal of Systems Integration*, Vol. 6, 1996, pp. 93-134.

[HARN99a] M. Harn, V. Berzins, and Luqi, "Software Evolution via Reusable Architecture", *Proceedings of 1999 IEEE Conference and Workshop on Engineering of Computer-Based Systems,* Nashville, Tennessee, March 7-12, 1999, pp. 11-17.

[HARN99b] M. Harn, "Computer-Aided Software Evolution Based on Inferred Dependencies", *Proceedings of Conference on Advanced Information Systems Engineering: 6th Doctoral Consortium,* Heidelberg, Germany, June 14-15, 1999, pp. 116-127.

[HARN99c] M. Harn, V. Berzins, and Luqi "A Dependency Computing Model for Software Evolution", *Proceedings of the Eleventh International Conference on Software Engineering and Knowledge Engineering*, Kaiserslautern, Germany, June 17-19, 1999, pp. 278-282.

[HARN99d] M. Harn, V. Berzins, Luqi, and W. Kemple, "Evolution of C4I Systems", *Proceedings of 1999 Command and Control Research and Technology Symposium*, United States Naval War College, Newport, Rhode Island, June 29 - July 1, 1999, pp.1361-1380.

[HARN99e] M. Harn, V. Berzins, and Luqi, "Computer-Aided Software Evolution Based on a Formal Model", *Proceedings of the Thirteenth International Conference on Systems Engineering*, Las Vegas, Nevada, August 9-12, 1999, pp. CS:55-60.

[HARN99f] M. Harn, "Computer-Aided Software Evolution System Based on Inferred Dependencies," *Ph.D. Dissertation*, Computer Science Department, Naval Postgraduate School, Monterey, CA, December, 1999.

[LIEB93] K. J. Lieberherr and C. Xiao, "Object-Oriented Software Evolution," *IEEE Trans. on Software Engineering*, Vol. 19, No. 4, April 1993, pp. 313-343.

[LUQI88a] Luqi and M. Ketabchi, "A Computer-Aided Prototyping System," *IEEE Software*, March 1988. pp. 66-72.

[LUQI88b] Luqi and V. Berzins, "Rapidly Prototyping Real-Time Systems," *IEEE Software*, September 1988. pp. 25-36.

[LUQI89] Luqi, "Software Evolution Through Rapid Prototyping," *IEEE Computer*, May 1989, pp. 13-25.

[LUQI90] Luqi, "A Graph Model for Software Evolution," *IEEE Trans. on Software Engineering*, Vol. 16, No. 8, August 1990, pp. 917-927.

[LUQI92] Luqi, "Computer-Aided Prototyping for a Command-And-Control System Using CAPS," *IEEE Software*, January 1992, pp. 56-67.

[LUQI93] Luqi, "How to Use Prototyping for Requirements Engineering," *Proceedings of IEEE/ACM Symposium on Requirements Engineering*, San Diego, CA, January 1993, p. 229.

[LUQI97] Luqi and J. A. Goguen, "Formal Methods Promises and Problems," *IEEE Software*, January 1997, pp. 73-85.

[ROET98] W. H. Roetzheim, "Formal Process Improvement - Ignore at Your Own Risk," *Trends in Software Engineering Process Management*, November 1998, http://www.marotz.com/journal/nov98/fpi.htm

[SEIT98] L. Seiter, J. Palsberg, K. Leiberherr (1998), "Evolution of Object Behavior Using Context Relations", *IEEE Trans. on Software Engineering*, Vol. 24, No. 1, January, pp. 79-92.

[SOMM96] I. Sommerville, *Software Engineering*, Addison-Wesley, 1996.

[STEI91] R. Steigerwald, Luqi, and J. McDowell, "CASE Tool for Reusable Software Component Storage and Retrieval in Rapid Prototyping," *Information and Software Technology*, England, Vol. 38, No. 9, November 1991, pp. 698-706.

[SUN96] Sun, "The Java™ Phenomenon," document available on-line at http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[SYMA97] Symantec Corporation, "Symantec VisualCafe, Database Edition, Version 3.0", 1997-1998.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste. 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Department of Computer Science 1
   Code CS
   Naval Postgraduate School
   Monterey, CA 93943

4. Software Engineering Program 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

5. STB-ARL 1
   115 O'keefe Building
   Georgia Institute of Technology
   Atlanta, GA 30332-0800

6. Space and Naval Warfare Systems Center, D40 1
   5356 Hull Street
   San Diego, CA 92152-5001

7. Space and Naval Warfare Systems Center, Code D42 1
   5356 Hull Street
   San Diego, CA 92152-5001

8. SSC SD Technical Library, Code D0274 1
   5356 Hull Street
   San Diego, CA 92152-5001

9. Dr. Valdis Berzins, Code CS/Bz 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

10. Prof. Luqi, Code CS/Lq 1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943

11. Dr. David Hislop 1
Army Research Office
4300 S. Miami Blvd.
Research Triangle Park, NC 22709-2211

12. Douglas Lange, Code D4223 1
Space and Naval Warfare Systems Center
53560 Hull Street
San Diego, CA 92152-5001

13. LTC Meng-Chyi Harn 1
P.O. Box 90046-17, CHUNG-HO
TAIPEI, TAIWAN, R.O.C.

14. Hanh Le, Code D4223 1
Space and Naval Warfare Systems Center
53560 Hull Street
San Diego, CA 92152-5001